ADAPTIVE OPTICAL LINEAR ALGEBRA PROCESSORS

Final Report on AFOSR Grant 84-0239

unclassified
SECURITY CLASSIFICATION OF THIS PAGE

ADA202996

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-1248
6a. NAME OF PERFORMING ORGANIZATION David Casasent Carnegie Mellon University Pittsburgh, PA 15213 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Dr. C. Lee Giles, AFOSR/NE, Building 410 Bolling AFB, 20332
6c. ADDRESS (City, State, and ZIP Code) Department of Electrical & Computer Engg. Center for Excellence in Optical Data Processing	7b. ADDRESS (City, State, and ZIP Code) Building 410 Bolling Air Force Base Washington, D.C. 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SAME as 7a 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-84-0239
8c. ADDRESS (City, State, and ZIP Code) SAME as 7b	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2305; TASK NO. B1; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Optical Systolic Array Processors for Adaptive Phased Array Radar
Adaptive Optical Linear Algebra Processors

12. PERSONAL AUTHOR(S)
David Casasent and B.V.K. Vijaya Kumar

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL	FROM 9/1/84 TO 10/31/88	November 15, 1988	150

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Adaptive phased array radar, Adaptive processors, Bit partitioning, Computational fluid dynamics applications, Finite element processors, frequency-multiplexed architectures, (continued on reverse)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The final report on research into adaptive optical linear algebra processors includes 3 processors. These include: a space integrating frequency-multiplexed processor, a hybrid space and time integrating processor, and a heterodyned linear analog processor. We also address number representation work using twos complement and negative base representation, plus fundamental new concepts such as matrix and bit partitioning. The applications addressed for these various systems include: the solution of linear and nonlinear algebraic equations, partial differential equations, computational fluid dynamics, finite element problems, adaptive signal processing (including adaptive phased array radar processing), and the potential for use in various adaptive neural etc. processing. We also include data to demonstrate and quantify that a high-accuracy [illegible] digital multiplication by analog convolution [illegible] optical architecture is superior to a digital processor.

[illegible]

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified/Unlimited
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. C. Lee Giles, AFOSR/NE	22b. TELEPHONE (Include Area Code) 202-767-4932 / 22c. OFFICE SYMBOL NE

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ADAPTIVE OPTICAL LINEAR ALGEBRA PROCESSORS

Final Report on

AFOSR Grant 84-0239

Submitted by:
Carnegie Mellon University
Department of Electrical and Computer Engineering
Center for Excellence in Optical Data Processing
Pittsburgh, PA 15213
Principal Investigators: David Casasent and B.V.K. Vijaya Kumar

Submitted to:
Air Force Office of Scientific Research
Building 410
Bolling Air Force Base
Washington, D.C. 20332
Attention: Dr. C. Lee Giles

November 1988

ADAPTIVE OPTICAL LINEAR ALGEBRA PROCESSORS

Final Report on
AFOSR Grant 84-0239

ABSTRACT

The final report on research into adaptive optical linear algebra processors includes 3 processors. These include: a space integrating frequency-multiplexed processor, a hybrid space and time ntegrating processor, and a heterodyned linear algebra processor. We also address number representation work using twos complement and negative base representation, plus fundamental new concepts such as matrix and bit partitioning.

ADAPTIVE OPTICAL LINEAR ALGEBRA PROCESSORS

Final Report on
AFOSR Grant 84-0239

CHAPTER 1. INTRODUCTION

In this final report on this grant, we highlight our recent results of the last year. The thrust of the work on this grant addressed optical linear algebra processors with attention to their laboratory realization. We have demonstrated and fabricated three optical processors in the laboratory. These include: (1) a space integrating and frequency multiplexed system, (2) a space and time integrating architecture, and (3) an analog heterodyned processor. We have demonstrated these systems on diverse applications: the solution of parabolic differential equations, finite element problems, computational fluid dynamics, and adaptive phased array radar. A new iterative Fourier transform processor concept was also addressed. Finally, we advance a new case for high-accuracy optical matrix processors versus digital versions of these systems.

Chapter 2 reviews our first system and lab data on its use in the solution of parabolic differential equations. Chapters 3 and 4 detail the final ac-coupled version of the second system fabricated and its use in finite-element problems. Chapters 5 and 6 address its use in the solution of problems in computational fluid dynamics and adaptive phased array radar. Chapter 7 demonstrates its use in bit-partitioning to achieve any desired accuracy with no increase in hardware. Chapters 8 and 9 present data and adaptive phased array radar. Chapter 11 advances new reasons why an optical high-accuracy matrix processor is superior to a digital realization.

This grant has successfully advanced many new theoretical results in new number representations, bit partitioning to achieve any desired accuracy, matrix partitioning to handle matrices of any size, and a wealth of new algorithms. It has also resulted in four new architectures, laboratory data on three of these, and real time laboratory optical solutions for five diverse applications in linear algebra. Chapter 12 lists the 33 journal and conference papers published under this grant, the 29 presentations given under it and the 6 student theses it supported. This represents excellent results and documentation of our research on this grant.

CHAPTER 2:

REAL-TIME OPTICAL LABORATORY SOLUTION OF PARABOLIC DIFFERENTIAL EQUATIONS

Real-time optical laboratory solution of parabolic differential equations

David Casasent and James Jackson

An optical laboratory matrix–vector processor is used to solve parabolic differential equations (the transient diffusion equation with two space variables and time) by an explicit algorithm. This includes optical matrix–vector nonbase-2 encoded laboratory data, the combination of nonbase-2 and frequency-multiplexed data on such processors, a high-accuracy optical laboratory solution of a partial differential equation, new data partitioning techniques, and a discussion of a multiprocessor optical matrix–vector architecture.

I. Introduction

Many optical matrix–vector processors have been described,[1] but few have been fabricated, and limited laboratory data on the use of these systems in practical engineering problems have been presented. Section II reviews the well-engineered optical laboratory architecture used in our present studies. Section III discusses our case study, the algorithm used, the partitioning employed, and the encoding used. Section IV presents laboratory data obtained. A summary and conclusion are then advanced in Sec. V. Optical matrix–vector processors represent the basic elements of many optical neural networks and adaptive processors[2] and are thus of considerable interest.

II. Optical Laboratory Matrix–Vector System

The optical processor considered is shown in Fig. 1. Its fabrication and operation have been discussed elsewhere,[3] and thus only a brief review of the system is given here. The system consists of several linear laser diode (LD) point modulator arrays at P_1. These are imaged onto an acoustooptic (AO) cell at P_2. The AO cell is fed with three frequency-multiplexed input signals. The 1-D vector data in each row at P_1 multiply the three vectors (frequency multiplexed) in the AO cell point by point, and the output integrating lens forms the sum of each point-by-point product [and hence the vector inner product (VIP) of the P_1 data and the P_2 data on separate output detectors at P_3]. We typically employ different LD rows at P_1 and the frequency-multiplexing at P_2 to represent bipolar and complex-valued data. New data are fed to P_1 and P_2 each bit time T_B. For the present laboratory system, $T_B = 250$ ns and five input LDs are used, although the system can support a much higher data rate and more LDs per row. With the P_1 data being an encoded representation of a number that is unchanged for the aperture time of the AO cell and with the input AO cell data being another encoded word, the P_3 output is the convolution of the bits of the two data words. By the digital multiplication by analog convolution (DMAC) algorithm,[4,5] this P_3 output correlation function is the high-accuracy product of the two corresponding input words (in mixed radix representation). This is the manner by which this processor achieves high-accuracy vector inner product operations. As noted elsewhere,[6] the DMAC algorithm can be used with vector data encoded in any base. Thus this architecture is quite versatile and has been optically realized in the laboratory.[3] In earlier work,[7] we demonstrated the laboratory performance of this system in the solution of a nonlinear matrix equation using the system in an analog mode and with an iterative algorithm employed to solve a system of linear algebraic equations. Here we consider its laboratory performance in an explicit solution of a parabolic partial differential equation (PDE). This present application requires that the system be operated in its high-accuracy mode on encoded data and thus that it employ the DMAC algorithm. Analog solutions to PDEs have been demonstrated on other optical architectures[8] but not to high accuracy and not with matrices with a high condition number.

When this work was done both authors were with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213; J. Jackson is now with Westinghouse Electric Corporation, Baltimore, Maryland.

Received 31 August 1987.
0003-6935/88/142922-04$02.00/0.

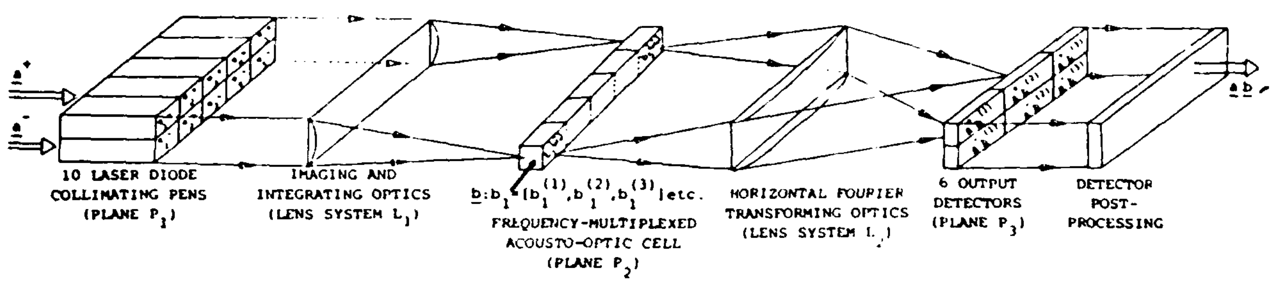

a
b
10 LASER DIODE COLLIMATING PENS (PLANE P_1)
IMAGING AND INTEGRATING OPTICS (LENS SYSTEM L_1)
$b:b_1^*(b_1^{(1)},b_1^{(2)},b_1^{(3)})$etc.
FREQUENCY-MULTIPLEXED ACOUSTO-OPTIC CELL (PLANE P_2)
HORIZONTAL FOURIER TRANSFORMING OPTICS (LENS SYSTEM L_2)
6 OUTPUT DETECTORS (PLANE P_3)
DETECTOR POST-PROCESSING
Ab

Fig. 1. Simplified schematic of the space integrating optical linear algebra processor.

III. Case Study

The problem we consider is the solution of the transient diffusion equation with two spatial variables plus time:

$$u_t = \alpha(u_{xx} + u_{yy}), \tag{1}$$

where subscripts denote partial derivatives with respect to time t or space (x or y), e.g., u_{xx} denotes the second partial derivative of u in x. We assume that the thermal diffusivity α is constant (as is typical of an isotropic time-variant medium), although the extension to the nonisotropic case is straightforward with α becoming a function of (x,y). Our purpose is to determine the 2-D temperature distribution u in x and y as a function of time t. This involves the calculation of $u(x,y,t_n)$ and its use to compute $u(x,y,t_{n+1})$ at the next time instant. Two types of problem formulation are possible, explicit matrix–vector and implicit linear algebraic equation solutions. From a study of both possibilities,[9] we selected the explicit solution, because the banded nature of the matrix is preserved and because a fixed number of calculations can be specified. We now detail our explicit problem formulation and solution.

A. Explicit Algorithm

We denote the space x,y indices by subscripts i,j and the time index by a superscript n, i.e., u_{ij}^{n+1} is $u[i\Delta x, j\Delta y, (n+1)\Delta t]$, where Δx, Δy, and Δt are the space and time step sizes used. We approximate u_t with a forward difference (forward Euler approximation) as

$$u_t = (u_{ij}^{n+1} - u_{ij}^n)/\Delta t. \tag{2}$$

A double central difference in x (index i) is used to approximate the second derivative u_{xx} as

$$u_{xx} = [u_{i+1,j}^n - 2u_{ij}^n + u_{i-1,j}^n]/(\Delta x)^2, \tag{3}$$

with a similar approximation used for u_{yy}. For the 1-D problem $u_t = \alpha u_{xx}$, the finite difference description becomes

$$u_i^{n+1} = \lambda u_{i+1}^n + (1 - 2\lambda)u_i^n + \lambda u_{i-1}^n, \tag{4}$$

where

$$\lambda = \alpha\Delta t/(\Delta x)^2. \tag{5}$$

This shows how the temperature at time step $n + 1$ depends explicitly on the prior temperature at time step n and constants. Denoting the spatial solution for all i at time step n as the vector $\mathbf{u}^n$, then

$$\mathbf{u}^{n+1} = \mathrm{A}\mathbf{u}^n, \tag{6}$$

where the matrix A is tridiagonal with all main diagonal elements being $(1 - 2\lambda)$ and with the elements directly above and below the main diagonal being λ.

For the 2-D problem in Eq. (1), the finite difference approximations yield

$$u_{ij}^{n+1} = \lambda u_{i+1,j}^n + \lambda u_{i-1,j}^n + \lambda u_{i,j+1}^n + \lambda u_{i,j-1}^n + (1 - 4\lambda)u_{ij}^n. \tag{7}$$

Using standard procedures,[10,11] we can describe Eq. (7) as a matrix–vector problem by ordering the N^2 elements of u_{ij} (assuming $N \times N$ spatial elements in x and y) at any time as an N^2-dimensional vector $\mathbf{u}$. With $\Delta x = \Delta y$, the solution for the spatial temperature distribution at time $n + 1$ is related to that at time n by an equation of the form of Eq. (6). However, the vector $\mathbf{u}$ and the matrix A are now of dimension N^2, and the matrix A now has multiple bands. Specifically, the central three diagonals are nonzero, and the diagonal elements $N - 1$ elements above and below the main diagonal are nonzero with all other elements being zero. In other partial differential equation problems and when higher-order difference approximations are used, more nonzero diagonals will occur $2N - 1$ elements from the main diagonal and the bandwidth of each diagonal band will increase. However, the basic multibanded matrix structure remains and is a characteristic feature of the matrix descriptions of many partial differential equation problems. Thus the general problem to be solved requires the matrix–vector multiplication in Eq. (6) for a multibanded matrix as described above. This matrix–vector multiplication must be performed to obtain $u(x,y)$ at each time step n and the results from the previous time step calculation used to compute the value of $u(x,y)$ at the next time step $n + 1$.

B. Case Study

The case study chosen was the solution of the 2-D diffusion equation for a 10- × 10-cm² aluminum plate with $\alpha = 0.86$ cm²/s. The plate was divided into $11 \times 11 = 121 = N^2$ square elements. The boundary conditions were zero temperature for the forty edge boundary points. For stability, we require $\lambda \leq 0.5$ and thus choose time steps $\Delta t = 0.29$ s. At each time step n, we require calculation of the temperature u in Eq. (6) for the $N^2 - 40 = 81$ internal spatial points (i,j). The specific A matrix, including boundary conditions, consists of $N \times N$ submatrices (each $N \times N$). The top left and bottom right submatrices are the identity matrices. The main diagonal submatrices are tridiagonal with all main diagonal elements equal to $\gamma = 1-4\lambda$ and with the diagonals one element above and below the main diagonal having entries $\lambda = \gamma\Delta t/(\Delta t)^2$. The matrices one block from the main diagonal are also diagonal matrices with all nonzero elements equal to λ. All other submatrices are all zero. Thus there are only five nonzero elements in any row of the $N^2 \times N^2$ matrix A. Our data flow and partitioning technique will exploit this matrix structure. We will consider the temperature output for forty time steps to demonstrate the explicit algorithm. We find that errors in the computed $\mathbf{u}^n$ at time step n will propagate into the calculated $\mathbf{u}^{n+1}$ temperature distribution. This process will continue, and errors will accumulate and make subseq ɹent results meaningless, unless we employ encode(data and the DMAC algorithm to achieve high accura :y.

C. Data Fl ɔw and Partitioning

At each time step n, the calculations in Eq. (6) require $N^2 - 40 = 81$ VIPs (ignoring the boundary elements). Since each row of A has only five nonzero elements, each VIP requires attention to at least five elements of A (as a minimum). The data flow and partitioning provided by our processor allow us to achieve each VIP using only the aforementioned minimum number of operations. Many architectures (both optical and digital and systolic arrays) cannot easily achieve such data flow. We now describe how the optical system of Fig. 1 can accommodate the proposed problem.

For the case of a general multiple-banded matrix problem, the architecture of Fig. 2, using multiple optical processors, is preferable. In this system, we represent the processor of Fig. 1 by the LD/AO/DET box shown. We employ diagonal partitioning and feed the central three nonzero diagonal elements of the main block diagonal matrix sequentially to three LDs in the first processor and the diagonal elements of the off-diagonal block submatrices to the LDs of subsequent processors in the cascade shown. The input vector data are delayed [by an amount, typically $(N - 1)T_B$, dependent on the block structure of the matrix A] and fed to the next processor in the cascade. The output vectors from each processor are collected and yield the final matrix–vector product. Thus, in this multiprocessor architecture, separate optical processors handle different matrix bands in the block matrix description. Data flow is ideal, with the general architecture being quite suitable for all multiple-banded matrix–vector problems. For the present problem, we would require a cascade of three such processors with no processor requiring more than three laser diode point modulators.

Although the architecture of Fig. 2 is quite general and versatile, for most finite difference and finite element problems, the number of nonzero matrix elements per row is quite modest, and a different data flow and partitioning technique can be used that re-

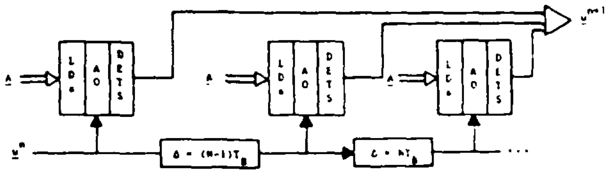


Fig. 2. Multiprocessor architecture for multiple banded matrices.

quires only one optical processor (with a modest number of input laser diodes). In this partitioning scheme, the nonzero matrix data A are fed to the AO cell, and the proper elements of the vector $\mathbf{u}$ are input to P_1 as required. This arrangement is also very attractive for data flow and all modest-sized problems. For the present case study problem, the data in each row of the matrix are the same, and thus the AO inputs at P_2 are a fixed set of five entries that are cyclically repeated. If α is not constant, the AO inputs change slowly with time, and this is easily achieved. Selection of the five elements of the vector $\mathbf{u}$ to the input to P_1 each T_B is easily achieved.[9] For example, the expression for the five $\mathbf{u}$ vector elements fed to P_1 at time n to calculate the matrix element $\mathbf{u}_{2,2}$ at time $n + 1$ is given by

$$u_{2,2}^{n+1} = \lambda u_{2,1}^n + \lambda u_{3,2}^n + (1 - 4\lambda)u_{2,2}^n + \lambda u_{2,3}^n + \lambda u_{1,2}^n. \qquad (8)$$

Extensions to the general element u_{ij}^{n+1} are easily obtained and allow optimal data flow. This was the partitioning and data flow we employed in our present laboratory tests. This partitioning arrangement allows us to solve a banded matrix problem whose size (121 × 121) exceeds that of the processor (five point modulator channels). Additional matrix bands can be handled similarly and partial matrix–vector products easily updated.

IV. Optical Laboratory Data

We first operated the optical laboratory system in the analog mode and found that after ten time steps, the error in the computed temperature distribution was unacceptable (being above 2%). We thus employed only encoded high-accuracy operation of the system. We used different bases B in the DMAC algorithm for the first time on this laboratory system. With $B = 5$ and with five digits of data, we achieve a data dynamic range of $(B - 1)^5 = 2^{15}$ or 15-bit accuracy. We ran the PDE problem on the optical system for forty time steps with the system operated in bases 2, 3, 4, and 5. In the first three tests, no errors were obtained, and performance was perfect. (Each test involved nearly 1.6 million multiplications and additions.) In tests with base 5, after $40\Delta t$, we obtained an average error of 0.02% in the calculated $u(x,y)$ distribution. This is attributed to component errors exceeding the separation between levels in the output A-D converter. Thus, for the present hardware, we are restricted to operate no higher than base 4. We then operated the system frequency-multiplexed with base 3 encoding and obtained perfect results again. This represents the first successful operation of this optical

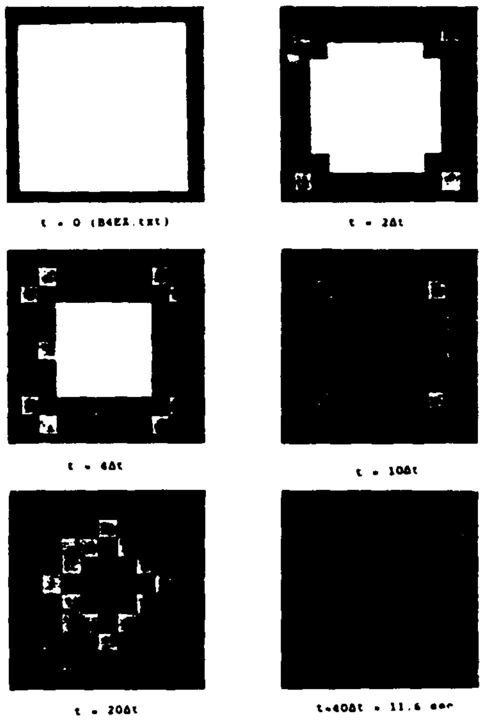


Fig. 3. Representative $u(x,y)$ temperature distribution outputs obtained at different time steps on the optical laboratory system of Fig. 1 operated with encoded data in base-4. The original photographs have the amplitude of the temperature distribution color-encoded.

laboratory system on encoded data frequency-multiplexed in the solution of a PDE.

To provide graphic output data from the optical system, we show the 2-D temperature distribution $u(x,y)$ calculated by the optical system at several time steps: $t = 0$, $t = 2\Delta t$, etc. up to $t = 40\Delta t$. These data are shown in Fig. 3. The original photographs have the temperature distribution color-encoded. However, the black and white data shown still indicate pictorially the performance and operation of the system and its successful calculation of the $u(x,y,t^n)$ temperature distributions with time.

V. Summary and Conclusion

This paper has detailed an explicit algorithm to solve parabolic differential equations, specifically the diffusion equation, for the temperature distribution with two spatial variables and one temporal variable. This paper represents new optical linear algebra laboratory data results in the solution of a practical engineering problem, specifically the use and demonstration of nonbase-2 encoded data, the solution of a PDE (the diffusion equation), and the use of nonbase-2 frequency-multiplexed data encoding. We also advanced and demonstrated (on a laboratory system) new partitioning techniques and how a multiple optical processor architecture could be used for solving multiple-banded matrix problems.

References

1. Special Issue on Optical Computing, Proc. IEEE **72**, (July 1984).
2. *Technical Diget, Topical Meeting on Optical Computing* (Optical Society of America, Washington, DC, 1987).
3. D. Casasent and J. Jackson, "Space and Frequency-Multiplexed Optical Linear Algebra Processors: Fabrication and Initial Tests," Appl. Opt. **25**, 2258 (1986).
4. H. J. Whitehouse and J. M. Speiser, "Linear Signal Processing Architectures," in *Aspects of Signal Processing, Part 2*, Proceedings, NATO Advanced Study Institute, G. Tacconi, Ed. (Reidel, Boston, 1976), pp. 669–702.
5. D. Psaltis, D. Casasent, D. Neft, and M. Carlotto, "Accurate Numerical Computation by Optical Convolution," Proc. Soc. Photo-Opt. Instrum. Eng. **232**, 151 (1980).
6. B. K. Taylor and D. Casasent, "Banded-Matrix High-Performance Algorithm and Architecture," Appl. Opt. **24**, 1476 (1985).
7. D. Casasent and J. Jackson, "Laboratory Optical Linear Algebra Processor for Optimal Control," Opt. Commun. **60**, 1 (15 Oct. 1986).
8. S. H. Lee, "Optical Analog Solutions of Partial Differential and Integral Equations," Opt. Eng. **24**, 41 (1985).
9. D. Casasent and J. Jackson, "Real-Time Optical Laboratory Linear Algebra Solution of Partial Differential Equations," Proc. Soc. Photo-Opt. Instrum. Eng. **698**, 000 (1986).
10. L. A. Hagemann and D. M. Young, *Applied Iterative Methods* (Academic, New York, 1981).
11. R. Vichnetvetsky, *Computer Methods for Differential Equations I* (Prentice-Hall, New York, 1981).

We acknowledge the support of the laboratory hardware for this system by the Air Force Office of Scientific Research (grant AFOSR-84-0239) and its use in the solution of partial differential equations by the National Science Foundation (grant ECS-8500830) and partial support by NASA Langley Research Center (grant NAG-1-575).

CHAPTER 3:

OPTICAL MATRIX-VECTOR LABORATORY DATA FOR FINITE ELEMENT PROBLEMS

SPIE Vol. 939
April 1988

Optical matrix-vector laboratory data for finite element problems

B.K. Taylor and D.P. Casasent
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, Pennslyvania 15213

ABSTRACT

We detail the use of an optical linear algebra processor for a finite element processing application. A linear time-varying structural mechanics finite element earthquake case study is described. The structure response under earthquake loading is considered, and the solution is obtained with the Newmark direct integration algorithm. The optical architecture for performing the required computationally burdensome banded matrix-vector operations is reviewed. The case study was solved on a laboratory version of the optical processor, and the results are presented.

1 INTRODUCTION

Finite element analysis is one of many scientific computing applications that require an enormous number of linear algebra (matrix-vector) computations. Many specialized optical linear algebra processors have been proposed to provide fast processing for many such applications [1,2]. This paper illustrates the use of a specific optical processor [3] that is well-suited to handle the computational tasks of a linear dynamic structural mechanics finite element analysis problem. The solution of a static structural mechanics finite element analysis has been described previously [4,5]. Few laboratory results from optical computing systems have been presented. Thus, our laboratory results with a reduced implementation of the processor as a proof-of-principle demonstration are quite unique. We first review the optical linear algebra processor architecture and discuss partioning and data representation issues (Section 2). Next, we present our finite element analysis case study formulation (Section 3), and the time-stepping Newmark solution algorithm that is used for the case study (Section 4). The laboratory system is described and the laboratory solution results are then examined (Section 5).

2 OPTICAL LINEAR ALGEBRA PROCESSOR

The optical linear algebra processor architecture that we consider in this paper is shown in Figure 1. The processor uses M laser diodes as point modulators at P_1. Each point modulator is imaged onto a corresponding vertical region of a multi-channel acousto-optic (AO) cell at P_2. The light distribution leaving P_2 is spatially integrated vertically onto a linear detector array at P_3. Each output detector is followed by an A/D converter and a shift/accumulator register to provide time-integration at P_3. This optical computing architecture was first described several years ago [3], and its initial laboratory implementation has been previously presented [6]. Thus, the following descriptions of the processor operation, partitioning, and data representation will be brief.

The optical linear algebra processor is capable of achieving high-accuracy optical matrix-vector operations as follows. We consider only one of the M vertical processor channels for multiplying two N-bit

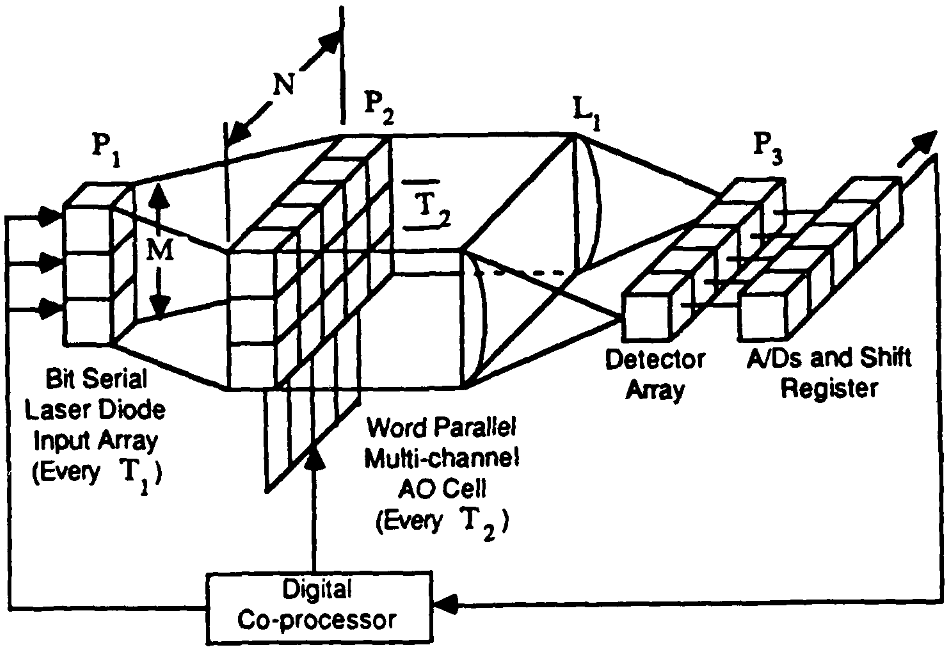

N
P2
L1
P1
P3
M
T2
Bit Serial Laser Diode Input Array (Every T1)
Word Parallel Multi-channel AO Cell (Every T2)
Detector Array
A/Ds and Shift Register
Digital Co-processor

Figure 1: Optical time- and space-integrating architecture

numbers. The digitally encoded bits of the multiplier are fed sequentially to the P_1 point modulator, and the bits of the multiplicand are fed word parallel to the AO cell at P_2. The word parallel multiplicand bits are present in one vertical processor channel section of the AO cell for a time T_2. During each T_2, the corresponding P_1 point modulator is pulsed on every T_1 with one of the bits of the multiplier. Each P_1 point modulator is pulsed on N times every T_2, thus $NT_1 = T_2$. Each T_1, one bit of the multiplier is multiplied by all N bits of the multiplicand and the output scalar-word product is imaged onto the N detectors at P_3. The P_3 data is shifted every T_1, and the scalar-word product for the next T_1 is formed and added to the prior (shifted) P_3 data. Thus, P_3 accumulates partial products of the multiplication result. After $T_2 = NT_1$, the P_3 output is the mixed radix representation of the product of the multiplier and multiplicand. When all M channels are considered, M scalar multiplications of different number pairs are performed every T_2 in the processor. Thus, an M-element vector inner product (VIP) is computed on the processor each T_2 by space integration. The mixed radix output is easily converted to conventional binary by A/D conversion and shift/adds of successive digits as they are shifted out of P_3 every T_1. This is known as the digital multiplication by analog convolution (DMAC) algorithm [7-9].

This architecture performs banded matrix vector products very efficiently [3]. In this algorithm, the contents of M diagonals of the matrix are fed to the M point modulators at P_1. For banded matrices, no out-of-band zero diagonals are processed. The vector elements are fed to the P_2 AO cell and reused in a systolic fashion in all M processor channels. Likewise, matrix partitioning [3] is easily achieved on this system when matrix bandwidths and vector lengths are larger than M. The matrix is then partitioned into diagonal ribbons, M diagonals are processed at one time, and the results for all diagonals are accumulated. With this partitioning technique, data flow and bookkeeping are simplified, and processor dead time is minimized.

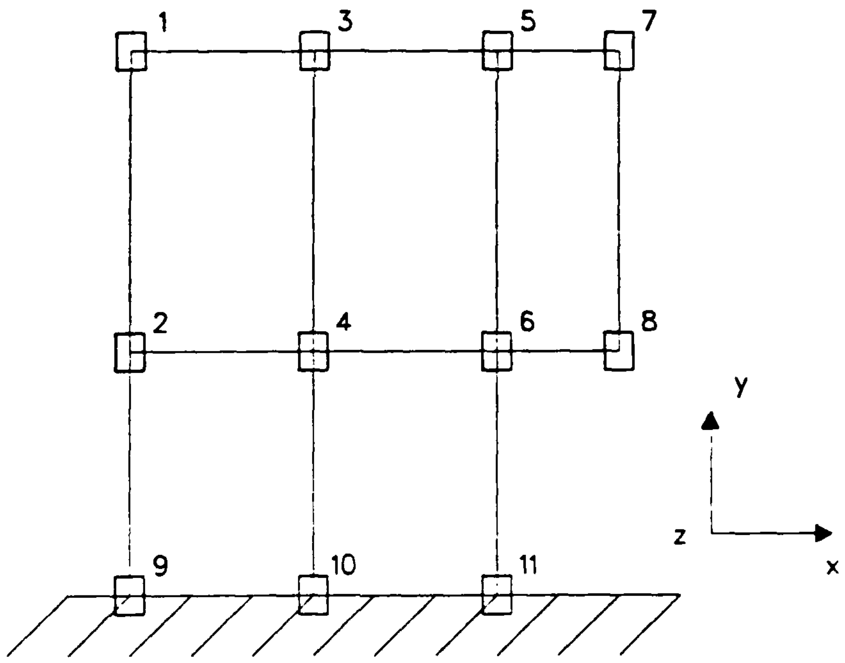


Figure 2: Case study structure finite element model

If the number of desired encoding bits N_b is larger than the number of available AO cell channels N at P_2 and P_3, bit partitioning [6] is also easily implemented. Since there are no carries in DMAC until mixed radix to binary conversion, we simply process the first N bits of the multiplicand, store the results, process the next N bits, etc. [6] Thus any desired accuracy may be achieved with only N channels at P_2 and P_3.

Many techniques for representing bipolar data on optical numerical processors have been presented [10-14] Our processor can be operated using a negative base representation for bipolar data [14]. This method handles negative numbers efficiently without requiring a significant number of extra bits, additional processing, or time or space multiplexing. Finally, we note that the DMAC algorithm is valid for data encoded in any radix [3]. The use of higher radices significantly increases computational throughput within A/D output P_3 converter limits.

3 FINITE ELEMENT CASE STUDY

This section details the case study using the notation that boldface uppercase letters represent square matrices, and boldface lowercase letters represent column vectors. The case study is a linear dynamic finite element structural mechanics problem. The plane frame structure model in Figure 2 is used, and the effects of simulated earthquake loadings applied to this structure are analyzed by our optical laboratory system. The structure is modeled with standard beam elements [15]. There are 11 nodes with 3 degrees-of-freedom at each node (displacement in x and y, and rotation about z).

Dynamic analysis [16] of the structure in Figure 2 requires solution of the matrix equation

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{p}(t), \tag{1}$$

where **M**, **C**, and **K** are the 33 × 33 mass, damping, and stiffness matrices, the vector $\mathbf{p}(t)$ is a vector of any externally applied time-varying loads, and $\ddot{\mathbf{d}}, \dot{\mathbf{d}}$ and **d** are the acceleration, velocity, and displacement vectors, respectively. The matrices **M**, **C**, and **K** describe the distribution of mass, damping, and elasticity throughout the structure, respectively. The formulas for both **K** and **M** may be found in standard references [15]. This formulation uses a consistent (distributed) mass matrix, as opposed to a lumped (diagonal) mass matrix. The Rayleigh damping formulation [16] with 2% damping is used to create the damping matrix **C**. With this formulation **C** is a linear combination of **M** and **K**. The nonzero entries of **K**, **C**, and **M** correspond to physical elastic, damping, and inertial coupling between nodes in the structure model. The nodes of finite element models are numbered to yield banded matrices. In our case study, the structure model nodes of Figure 2 are numbered to minimize the bandwidths of **K**, **C**, and **M** to 21. We consider a linear analysis, i.e. the mass, damping, and stiffness matrices remain constant throughout the problem solution.

Equation 1 is a system of second order linear differential equations that must be solved for the unknown vectors $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and **d**. The particular solution algorithm that we use is detailed in Section 4. We now detail more specific problem formulation issues for our earthquake analysis.

The purpose of this case study is to provide an analysis problem that is representative of similar larger scientific computing tasks. This finite element earthquake analysis is an example of one such problem. As with many earthquake analysis efforts, the earthquake acceleration, velocity, and displacement data are artificially generated [17,18].

In our dynamic analysis, we investigate the response of this structure to earthquake loadings. In such an analysis, the ground nodes (9,10,11) cannot be constrained, as the earthquake imparts forces (part of $\mathbf{p}(t)$) that cause displacements, velocities, and accelerations at those nodes. Instead, the time-histories of the displacements, velocities, and accelerations of the ground nodes are prescribed from the earthquake data. Thus, we specify $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and **d** at nodes 9-11. From this information, the movements of the other nodes in the structure can be calculated as we will detail. Simulated horizontal (x) acceleration, velocity, and displacement data for nodes 9-11 were generated for a 10 second tremor, typical of a strong ground motion earthquake.

We separate equation 1 into two equations with partitioned vectors denoted by the subscripts 1 and 2, i.e.

$$\begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{d}}_1 \\ \ddot{\mathbf{d}}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{d}}_1 \\ \dot{\mathbf{d}}_2 \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{Bmatrix}, \tag{2}$$

where the matrices have been similarly partitioned. The first equation in (2) can be rearranged as

$$\mathbf{M}_{11}\ddot{\mathbf{d}}_1 + \mathbf{C}_{11}\dot{\mathbf{d}}_1 + \mathbf{K}_{11}\mathbf{d}_1 = \mathbf{p}_1 - \mathbf{M}_{12}\ddot{\mathbf{d}}_2 - \mathbf{C}_{12}\dot{\mathbf{d}}_2 - \mathbf{K}_{12}\mathbf{d}_2, \tag{3}$$

where the right-hand side is known, and the matrices are 24 × 24 (with a bandwidth of 21) and the vectors are 24 × 1. The top vector partition (subscript 1) includes nodes 1-8 where $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and **d** are unknown and must be determined. For these nodes (in our case study) the load vector $\mathbf{p}_1 = \mathbf{0}$ since the earthquake loading is at nodes 9-11. The second partition (subscript 2) includes nodes 9-11 where $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and **d** are known and specified. Our concern is to solve equation (3) for $\ddot{\mathbf{d}}_1$, $\dot{\mathbf{d}}_1$, and $\mathbf{d}_1$. Once these values are obtained, we can calculate $\mathbf{p}_2$ (the part of $\mathbf{p}(t)$ due to the earthquake forces for nodes 9-11) from the second equation in (2) if desired. In general, we solve for an accurate time-history of these nodal displacements during the earthquake loading. The particular quantities of concern are usually the maximum displacements of the nodes during the earthquake and their final displacements.

This matrix equation (3), which has the same form as (1), must be solved for the acceleration, velocity, and displacement vectors $\ddot{\mathbf{d}}_1$, $\dot{\mathbf{d}}_1$, and $\mathbf{d}_1$. We now detail the time-integrating solution method we use to solve (3).

4 SOLUTION ALGORITHM

Equation (3) will be solved for $\ddot{\mathbf{d}}_1$, $\dot{\mathbf{d}}_1$, and $\mathbf{d}_1$ using the Newmark direct integration method [16]. Equation (3) has the same form as equation (1), and the notation in (1) will be used in this Section. The subscripts of $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and $\mathbf{d}$ will denote values for a particular time step. The Newmark integration scheme uses the following approximations (which assume a linear acceleration between time steps):

$$\dot{\mathbf{d}}_{t+\Delta t} = \dot{\mathbf{d}}_t + [(1-\delta)\ddot{\mathbf{d}}_t + \delta\ddot{\mathbf{d}}_{t+\Delta t}]\Delta t \tag{4}$$

$$\mathbf{d}_{t+\Delta t} = \mathbf{d}_t + \dot{\mathbf{d}}_t\Delta t + [(0.5-\alpha)\ddot{\mathbf{d}}_t + \alpha\ddot{\mathbf{d}}_{t+\Delta t}]\Delta t^2. \tag{5}$$

Equations (4) and (5) are used to relate $\ddot{\mathbf{d}}_{t+\Delta t}$ and $\dot{\mathbf{d}}_{t+\Delta t}$ to $\mathbf{d}_{t+\Delta t}$ and the previous values of $\ddot{\mathbf{d}}_t$, $\dot{\mathbf{d}}_t$, and $\mathbf{d}_t$. The integration constants α and δ control the accuracy and stability of the integration scheme (as detailed later), and Δt is the time step used for the integration.

When (4) and (5) are substituted into (1) at time $t+\Delta t$, we obtain

$$\hat{\mathbf{K}}\mathbf{d}_{t+\Delta t} = \hat{\mathbf{p}}_{t+\Delta t}, \tag{6}$$

where $\hat{\mathbf{K}}$ is the effective stiffness matrix

$$\hat{\mathbf{K}} = \mathbf{K} + a_0\mathbf{M} + a_1\mathbf{C}, \tag{7}$$

and the right hand side vector in (6) is

$$\hat{\mathbf{p}}_{t+\Delta t} = \mathbf{p}_{t+\Delta t} + \mathbf{M}(a_0\mathbf{d}_t + a_2\dot{\mathbf{d}}_t + a_3\ddot{\mathbf{d}}_t) + \mathbf{C}(a_1\mathbf{d}_t + a_4\dot{\mathbf{d}}_t + a_5\ddot{\mathbf{d}}_t), \tag{8}$$

where

$$\begin{array}{ll} a_0 = \frac{1}{\alpha\Delta t^2} & a_1 = \frac{\delta}{\alpha\Delta t} \\ a_2 = \frac{1}{\alpha\Delta t} & a_3 = \frac{1}{2\alpha} - 1 \\ a_4 = \frac{\delta}{\alpha} - 1 & a_5 = (\frac{\Delta t}{2})(\frac{\delta}{\alpha} - 2) \\ a_6 = \Delta t(1-\delta) & a_7 = \delta\Delta t. \end{array}$$

For δ and α we choose

$$\delta \geq 0.50 \qquad \alpha \geq 0.25(0.5+\delta)^2.$$

These conditions insure unconditional stability. With the selection of $\delta = 0.50$ and $\alpha = 0.25$ we obtain the best integration accuracy [16].

For this analysis we are interested in accurately determining the response of the fundamental oscillatory mode of the structure, which is 3.64 Hz. The time step $\Delta t = 1/60 sec.$ is chosen for the Newmark algorithm to be small enough to sufficiently sample the 3.64 Hz fundamental oscillation (about 7 ×

Nyquist). For a 10 second analysis, 600 iterations of the Newmark algorithm are used, and thus equation 8 is solved 600 times.

To determine $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and $\mathbf{d}$ for every time step, we initialize the values of $\ddot{\mathbf{d}}_0$, $\dot{\mathbf{d}}_0$, and $\mathbf{d}_0$ to zero for our case study. We then calculate $\hat{\mathbf{p}}_{t+\Delta t}$ from (8) and then solve the linear algebraic equation in (6) for $\mathbf{d}_{t+\Delta t}$. From this we find $\ddot{\mathbf{d}}_{t+\Delta t}$ from

$$\ddot{\mathbf{d}}_{t+\Delta t} = a_0(\mathbf{d}_{t+\Delta t} - \mathbf{d}_t) - a_2\dot{\mathbf{d}}_t - a_3\ddot{\mathbf{d}}_t \tag{9}$$

which is (5) rearranged, and then $\dot{\mathbf{d}}_{t+\Delta t}$ from

$$\dot{\mathbf{d}}_{t+\Delta t} = \dot{\mathbf{d}}_t + a_6\ddot{\mathbf{d}}_t + a_7\ddot{\mathbf{d}}_{t+\Delta t}, \tag{10}$$

which is (4) written in terms of a_n.

Equations (6) - (10) result from the Newmark algorithm. Since $\hat{\mathbf{K}}$ is constant during the solution, we factorize it as $\hat{\mathbf{K}} = \mathbf{L}\mathbf{L}^{\mathbf{T}}$, and thus solve (6) for $\mathbf{d}_{t+\Delta t}$ by

$$\mathbf{d}_{t+\Delta t} = (\mathbf{L}\mathbf{L}^{\mathbf{T}})^{-1}\hat{\mathbf{p}}_{t+\Delta t}. \tag{11}$$

Equation (11) is only a matrix-vector multiplication. The most time consuming computational step is evaluating (8), which requires banded matrix-vector multiplications and of order $2N^2$ operations per iteration. In our optical laboratory tests we evaluated (8) optically.

5 LABORATORY SYSTEM AND RESULTS

We now describe the optical linear algebra system used to solve the case study in the laboratory. For initial investigation and as a proof-of-principle demonstration, we implement a reduced M=1 and N=1 channel version of the processor. The partitioning methods described in Section 2 are used for processing the data, and the encoding radix for the case study solution is binary. The numeric dynamic range of the case study is approximately 10^{10}. We use $N_a = 48$ bit encoding which covers the numeric dynamic range ($2^{48} \approx 10^{14}$) and yields accurate solution results. In this specific implementation, bipolar numbers are used and are encoded with a sign-magnitude representation. This is possible in an M=1 system as documented elsewhere [3].

A photograph of the laboratory system is shown in Figure 3. A 30 mW laser diode is used as the point modulator at P_1, emitting at a wavelength of 780 nm. The laser diode package and its associated drive and modulation circuitry are mounted on a board approximately two inches square. The divergent laser diode beam is collimated by an $f_L = 4.5mm$ spherical lens and focused at P_2 with an $f_L = 100mm$ cylindrical lens. A 32-channel acousto-optic cell is used at P_2. The AO cell has a 300 MHz center frequency and a 100 MHz modulation bandwidth. The diffraction efficiency of the cell is 60%/Watt. The first order diffracted light output leaving P_2 is focused onto a single channel detector/amplifier at P_3. A silicon p-i-n detector is mounted at the input of a high-gain transimpedance amplifier. The frequency response of the amplifier rolls off 3 dB near 500 MHz. The output of the amplifier is demodulated and is fed to the A/D and shift/accumulate circuitry.

Data is input to the laboratory optical processor and collected from the output of the optical processor at high-speed (10 Mbit/sec) by a digital support system through high-speed memory boards. The digital support system is described elsewhere [6].

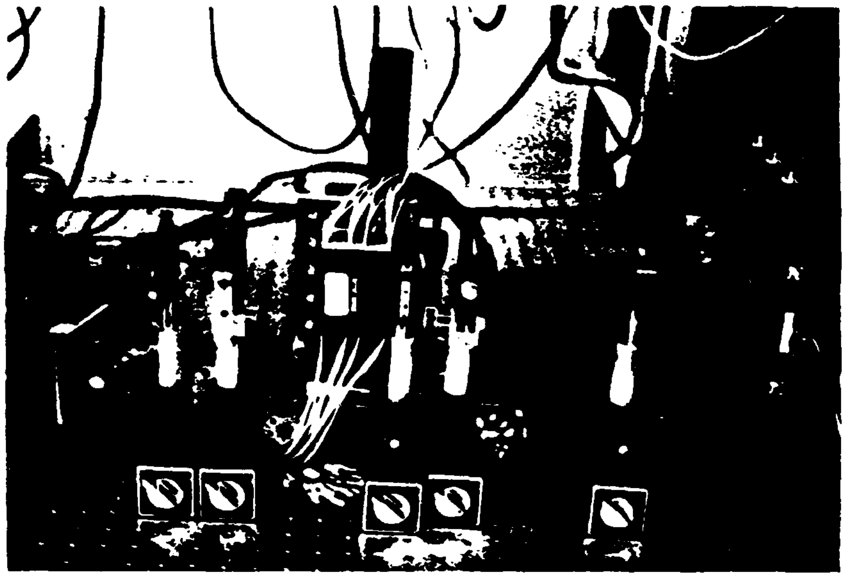


Figure 3: Laboratory optical linear algebra processor

In previous laboratory implementations, thermal drift effects at P_3 limited accurate processor operation to a few hours, at which time one had to compensate for the drift. In these previous implementations, the output from P_2 was detected at P_3 and DC-coupled into the amplifier, where a direct detection of the output level was used. Drift in the output levels occurred as a result of ambient temperature changes in the laboratory, and localized heating effects of the integrated detector/preamplifier. In order to eliminate any P_3 drift effects, an AC-coupled modulation scheme was employed for the laboratory optical processor implementation discussed in this paper.

We now describe the AC-coupled modulation scheme. The laser diode at P_1 is biased at an operating point near the middle of the linear portion of its operating curve. A constant light output at the bias point represents a zero for data encoding. To represent a nonzero bit, the laser diode is modulated about the bias point by modulating the amplitude of a high frequency carrier (approximately 400 MHz). The carrier frequency must be significantly higher than the 10 Mbit/sec modulation data rate. To represent bits of different data encoding levels, the carrier is simply amplitude modulated corresponding to the bit levels (thus no carrier indicates a zero bit). For the present lab data binary encoding is used. The AO cell at P_2 is fed with a signal at 300 MHz (the center frequency of the AO cell) and modulated by turning the carrier on at different levels, and off for a zero data bit. The light distribution leaving P_2 representing any nonzero products will thus have temporal data modulation at the P_1 carrier frequency. The detector/amplifier output from P_3 is AC-coupled into a demodulator circuit. The input to the demodulator is now unaffected by any low frequency thermal drift at P_3. The demodulator circuit is an envelope detector which consists of an RF transformer, RF diode, and low pass filter. This AC-coupled modulation technique has been extremely successful in the laboratory. The optical processor operates without any problems from thermal drift at P_3. A mixer circuit may also be used for demodulation. In this case the P_1 carrier is used as the local oscillator, the detector/amplifier output is input to the mixer

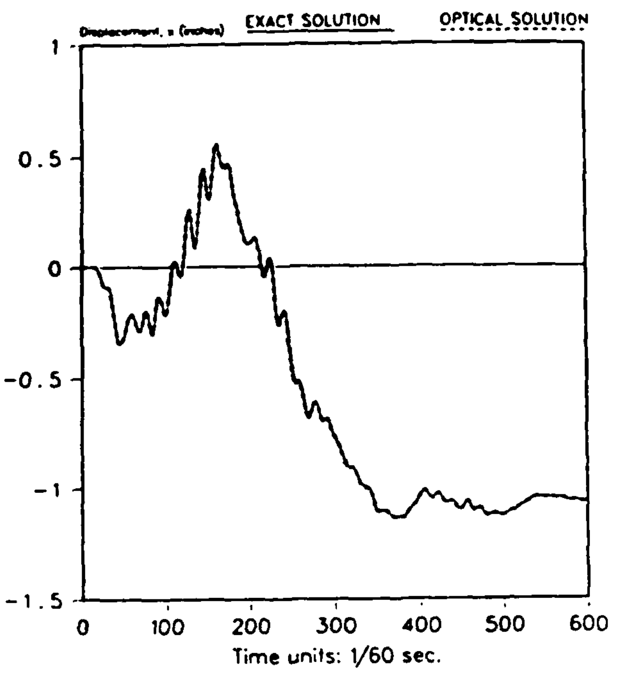


Figure 4: Case study solution comparison

RF port, and the demodulated signal is present at the IF port of the mixer. This method has also been used in the laboratory with equal success, however it requires the additional local oscillator reference line.

The finite element case study solved with the Newmark algorithm was implemented on the laboratory optical processor described above. The response of the x coordinate displacement of node 1 in Figure 2 is used to compare the 48-bit digital solution with the laboratory optical solution of the case study. The solution results are illustrated in Figure 4. The horizontal displacement in inches is plotted on the vertical axis vs. the 600 time steps of the 10 second tremor on the horizontal axis. The solid line indicates the 48-bit digital solution, and the dashed line indicates the N_b = 48 bit optical laboratory solution. Both solutions are nearly identical with an average percent error of less than 1%. This value is an average over all 600 points of the absolute percent error between the laboratory optical and digital solutions. The standard deviation of the percent error is also less than 1%.

6 DISCUSSION

The small errors that occurred in the laboratory optical processor results were not entirely unexpected for an initial laboratory system implementation. The source of the errors was discovered after additional testing of the laboratory optical processor to be the detector/amplifier used at P_3. In the AC-coupled modulation technique, the P_1 laser diode is biased on and thus emits a constant output (with no carrier) with no data input. Thus, first order diffracted light from P_2 will be incident on the detector at P_3 whenever nonzero data is input to the AO cell at P_2 (even with no P_1 data present). When nonzero data is present at both P_1 and P_2, the P_3 output signal is at the 100 MHz P_1 carrier frequency, otherwise it is at DC. This results in a low frequency variation (determined by the AO cell data, number of consecutive nonzero numbers, etc.) in the optical power incident on the detector. This low frequency variation affected the high frequency (400 MHz) gain of the the P_3 detector/amplifier. Thus, the output levels of the detector/amplifier and the demodulator were dependent on low frequency duty cycle variations of the data input to the AO cell. This problem can easily be corrected with proper coupling between the detector and the amplifier (which was not an alternative for our device), or with proper amplifier design. We are currently finishing fabrication and testing of a 10-channel detector/amplifier array to be used at P_3. Tests of the first operational channels verified the success of this new design, and that the previous detector problems are now absent in this new device. Initial tests of the optical system with this new detector showed no errors for various sets of test patterns of data. We plan to implement the case study on our laboratory optical system with the new detector/amplifier array as soon as the system fabrication is completed. We expect the optical processing results to coincide exactly with the digital results.

7 CONCLUSION

This paper has detailed the use of a laboratory optical linear algebra processing system to solve a linear dynamic finite element case study. The computationally-intensive banded matrix-vector products of the Newmark time-integration algorithm were implemented on the optical processor. Excellent laboratory results were obtained, and further improvement to digital accuracy can be achieved with a simple design modification.

References

[1] "Special issue on optical computing," *Proceedings IEEE*, Vol. 72, No. 7, July, 1984.

[2] D.P.Casasent, "Acoustooptic linear algebra processors: architectures, algorithms, and applications," *Proc. of IEEE*, vol. 72, pp. 831–849, July 1984.

[3] D.P.Casasent and B.K.Taylor, "Banded-matrix high-performance algorithm and architecture," *Applied Optics*, vol. 24, pp. 1476–1480, 15 May 1985.

[4] D. Casasent and B. K. Taylor, "Optical finite element processor," *Proc. SPIE*, vol. 564, pp. 139–149, August 1985.

[5] B.K.Taylor and D.P.Casasent, "Error-source effects in a high-accuracy optical finite-element processor," *Applied Optics*, vol. 25, pp. 966–975, 15 March 1986.

[6] D. Casasent and S. Riedl, "Time and space integrating optical laboratory matrix-vector array processor," *Proc. SPIE*, vol. 698, pp. 151–156, August 1986.

[7] E. Swartzlander, "The quasi-serial multiplier," *IEEE Trans. on Comp.*, vol. C-22, pp. 317–321, April 1973.

[8] H. Whitehouse and J. Speiser, "Linear signal processing architectures," in *Aspects of Signal Processing - Part II*, (G. Tacconi, ed.), (Boston, MA), pp. 669–702, NATO Advanced Study Institute, 1976.

[9] D. Psaltis, D. Casasent, D. Neft, and M. Carlotto, "Accurate numerical computation by optical convolution," *Proc. SPIE*, vol. 232, pp. 151–156, April 1980.

[10] J. Goodman, et al., "Parallel incoherent optical vector-matrix multiplier," BMD Technical Report No. L-723-1, February 1979.

[11] D.P.Casasent, J.Jackson, and C.P.Neuman, "Frequency-multiplexed and pipelined iterative optical systolic array processors," *Applied Optics*, vol. 22, pp. 115–124, 1 January 1983.

[12] R.P. Bocker, S.R. Clayton, and K. Bromley, "Electrooptical matrix multiplication using the twos complement arithmetic for improved accuracy," *Applied Optics*, vol. 23, pp. 2019–2021, July 1983.

[13] B.K.Taylor and D.P.Casasent, "Twos-complement data processing for improved encoded matrix-vector processors," *Applied Optics*, vol. 25, pp. 956–961, 15 March 1986.

[14] C.Perlee and D.P.Casasent, "Negative base encoding in optical linear algebra processors," *Applied Optics*, vol. 25, pp. 168–169, 15 January 1986.

[15] J. S. Przemieniecki, *Theory of Matrix Structural Analysis*. New York: McGraw-Hill, 1968.

[16] K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1976.

[17] N. M. Newmark and E. Rosenblueth, *Fundamentals of Earthquake Engineering*. Prentice-Hall, Inc., 1971.

[18] M. Wakabayashi, *Design of Earthquake-Resistant Buildings*. McGraw-Hill, 1986.

CHAPTER 4:

OPTICAL LABORATORY SOLUTION AND ERROR MODEL SIMULATION OF A LINEAR TIME-VARYING FINITE ELEMENT SOLUTION

Proc. SPIE
Volume 977
August 1988

Optical Laboratory Solution and Error Model Simulation of a Linear Time-Varying Finite Element Equation

B.K. Taylor and D.P. Casasent
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, Pennslyvania 15213

ABSTRACT

We detail the use of simplified error models to accurately simulate and evaluate the performance of an optical linear algebra processor. The optical architecture used to perform banded matrix-vector products is reviewed along with the linear dynamic finite element case study. The laboratory hardware and AC-modulation technique used are presented. The individual processor error source models and their simulator implementation are detailed. Several significant simplifications are introduced to ease the computational requirements and complexity of the simulations. The error models are verified with a laboratory implementation of the processor, and are used to evaluate its potential performance.

1. INTRODUCTION

This paper describes the use of simplified error source models to accurately simulate an optical linear algebra processor. The error source models are general enough that they may be adapted to simulate a wide range of optical data processing architectures. Many specialized optical linear algebra processors have been proposed to provide fast processing for linear algebra applications [1,2]. This paper illustrates the use of a specific optical processor [3] that is well-suited to handle the computational tasks of a linear dynamic structural mechanics finite element analysis problem, specifically, the parallel computation of banded matrix-vector products. The solutions of static and dynamic structural mechanics finite element analysis problems have been described previously [4-6], along with initial laboratory results on a cross-section of the processor [6]. We first review the optical linear algebra processor architecture and discuss partioning and data representation issues (Section 2). Next, we present our finite element analysis case study (Section 3), and the AC-coupled modulation scheme and hardware details (Section 4). The error source models and simulations are presented (Section 5), and the laboratory processing results are then examined (Section 6).

2. OPTICAL LINEAR ALGEBRA PROCESSOR

The optical linear algebra processor architecture that we consider is shown in Figure 1. The processor uses M laser diodes as point modulators at P_1. Each point modulator is imaged onto a corresponding vertical region of a multi-channel acousto-optic (AO) cell at P_2. The light distribution leaving P_2 is spatially integrated vertically onto a linear detector array at P_3. Each output detector is followed by an A/D converter and a shift/accumulator register to provide time-integration at P_3. This emulates a high-speed shift register digital output system. This optical computing architecture was first described several years ago [3], and its initial laboratory implementations have been previously presented [6,7], thus, the following descriptions of the processor operation, partitioning, and data representation will be brief.

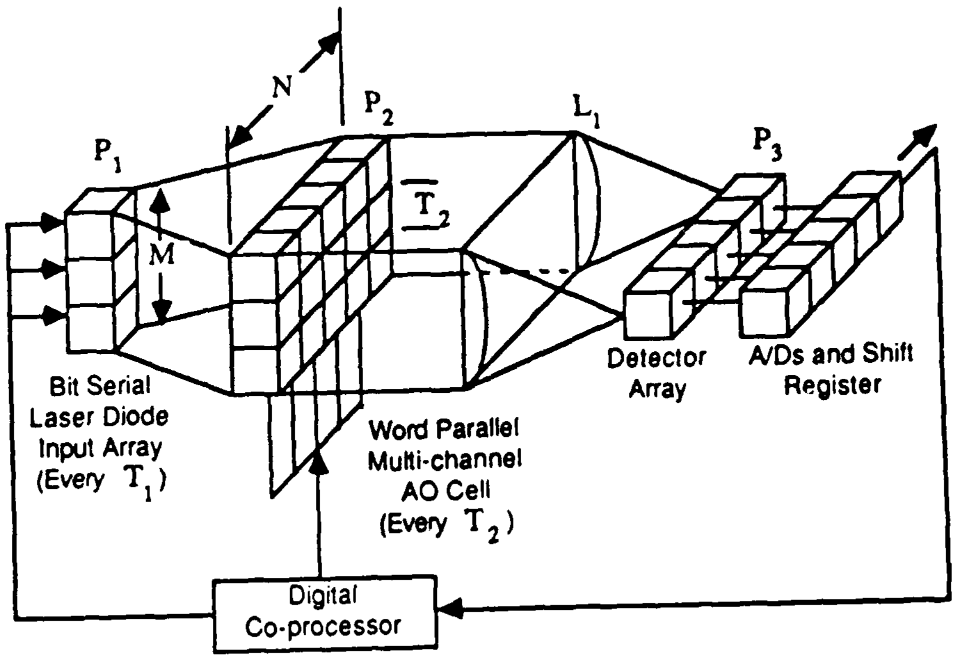

N
P2
L1
P1
P3
T2
M
Bit Serial Laser Diode Input Array (Every T1)
Word Parallel Multi-channel AO Cell (Every T2)
Detector Array
A/Ds and Shift Register
Digital Co-processor

Figure 1: Optical time- and space-integrating architecture.

The optical linear algebra processor is capable of achieving high-accuracy optical matrix-vector operations as follows. We consider only one of the M vertical processor channels for multiplying two N-bit numbers. The digitally encoded bits of the multiplier are fed sequentially to the P_1 point modulator, and the bits of the multiplicand are fed word parallel to the AO cell at P_2. The word parallel multiplicand bits are present in one vertical processor channel section of the AO cell for a time T_2. During each T_2, the corresponding P_1 point modulator is pulsed on every T_1 with one of the bits of the multiplier. Each P_1 point modulator is pulsed on N times every T_2, thus $NT_1 = T_2$. Each T_1, one bit of the multiplier is multiplied by all N bits of the multiplicand and the output scalar-word product is imaged onto the N detectors at P_3. The P_3 data is shifted every T_1, and the scalar-word product for the next T_1 is formed and added to the prior (shifted) P_3 data. Thus, P_3 accumulates partial products of the multiplication result. After $T_2 = NT_1$, the P_3 output is the mixed radix representation of the product of the multiplier and multiplicand. When all M channels are considered, M scalar multiplications and additions of different number pairs are performed every T_2 in the processor. Thus, an M-element vector inner product (VIP) is computed on the processor each T_2 by space integration. The mixed radix output is easily converted to conventional binary by A/D conversion and shift/adds of successive digits as they are shifted out of P_3 every T_1. This is known as the digital multiplication by analog convolution (DMAC) algorithm [8-10].

This architecture performs banded matrix vector products very efficiently [3]. In this algorithm, the contents of M diagonals of the matrix are fed to the M point modulators at P_1. For banded matrices, no out-of-band zero diagonals are processed. The vector elements are fed to the P_2 AO cell and reused in a systolic fashion in all M processor channels. Likewise, matrix partitioning [3] is easily achieved on this system when matrix bandwidths and vector lengths are larger than M. The matrix is partitioned into diagonal ribbons, M diagonals are processed at one time, and the results for all diagonals are accumulated. With this partitioning technique, data flow and bookkeeping are simplified, and processor dead time is minimized.

If the number of desired encoding bits N_a is larger than the number of available AO cell channels N at P_2 and P_3, bit partitioning [7] is also easily implemented. Since there are no carries in DMAC until mixed radix to binary conversion, we simply process the first N bits of the multiplicand, store the results, process the next N bits, etc. [7] Thus any desired accuracy may be achieved with only N channels at P_2 and P_3.

Many techniques for representing bipolar data on optical numerical processors have been presented [11-15] Our processor can be operated using a negative base representation for bipolar data [15]. This method handles negative numbers efficiently without requiring a significant number of extra bits, additional processing, or time or space multiplexing. Finally, we note that the DMAC algorithm is valid for data encoded in any radix [3]. The use of higher radices significantly increases computational throughput within A/D output P_3 converter limits.

3. LABORATORY SYSTEM

We now describe the optical linear algebra system used to solve the case study in the laboratory and evaluated the error source modeling. We implement a reduced M=1 and N=1 channel version of the processor. The partitioning methods described in Section 2 are used for processing the data, and the encoding radix for the case study solution is binary. The numeric dynamic range of the case study is approximately 10^{10}. We use $N_a = 48$ bit encoding which covers the numeric dynamic range ($2^{48} \approx 10^{14}$) and yields accurate solution results. In this specific implementation, bipolar numbers are used and are encoded with a sign-magnitude representation. This is possible in an M=1 system as documented elsewhere [3].

A photograph of the laboratory system is shown in Figure 2. A Sharp VSIS (V-channeled Substrate Inner Stripe) [16] 30 mW laser diode is used as the point modulator at P_1, emitting at a wavelength of 780 nm. The laser diode package and its associated drive and modulation circuitry are mounted on a board approximately two inches square. The divergent laser diode beam is collimated by an $f_L = 4.5$ mm spherical lens and focused at P_2 with an $f_L = 100$ mm cylindrical lens. A 32-channel acousto-optic cell is used at P_2. The AO cell has a 300 MHz center frequency and a 100 MHz modulation bandwidth. The diffraction efficiency of the cell is 60%/Watt. The first order diffracted light output leaving P_2 is focused onto a single channel of a 10-channel detector/amplifier array at P_3. A custom United Detector Technology array of 10 silicon p-i-n detectors feeds individual high-gain, low input impedance microwave amplifier chains. The bandpass of the amplifiers is 250 MHz to 450 MHz. The output of the amplifier is demodulated and is fed to the A/D and shift/accumulate circuitry.

Data is input to the laboratory optical processor and collected from the output of the optical processor at high-speed (10 Mbit/sec) by a digital support system through high-speed memory boards. The digital support system is described elsewhere [7].

In previous laboratory implementations, thermal drift effects at P_3 limited accurate processor operation to a few hours, at which time drift compensation was needed. In these previous implementations, the output from P_2 was detected at P_3 and DC-coupled into the amplifier, where a direct detection of the output level was used. Drift in the output levels occurred as a result of ambient temperature changes in the laboratory, and localized heating effects of the integrated detector/preamplifier. In order to eliminate any P_3 drift effects, an AC-coupled modulation scheme was employed for the laboratory optical processor implementation discussed in this paper.

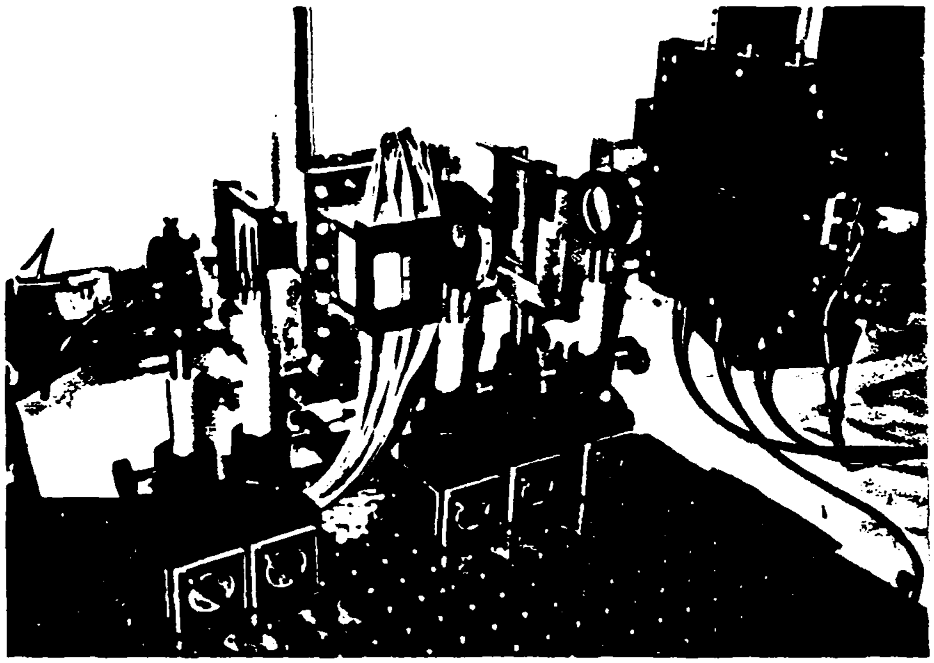


Figure 2: Laboratory optical linear algebra processor.

We now describe the AC-coupled modulation scheme. The laser diode at P_1 is biased at an operating point near the middle of the linear portion of its operating curve. A constant light output at the bias point represents a zero for data encoding. To represent a nonzero bit, the laser diode is modulated about the bias point by modulating the amplitude of a high frequency 400 MHz carrier. The carrier frequency must be significantly higher than the 10 Mbit/sec modulation data rate. To represent bits with different data encoding levels, the carrier is simply amplitude modulated (by varying the modulation depth) corresponding to the bit levels (with no carrier indicating a zero bit). For the present lab data, binary encoding is used. This maintains the output of the laser diode constant using a monitor photodiode feedback circuit. The AO cell at P_2 is fed with a signal at 300 MHz (the center frequency of the AO cell) and is modulated by turning the carrier on at different levels and off for a zero data bit. The light distribution leaving P_2 representing any nonzero products will thus have temporal data modulation at the P_1 carrier frequency. The detector/amplifier output from P_3 is AC-coupled into a demodulator circuit. The input to the demodulator is now unaffected by any low frequency thermal drift at P_3. The demodulator circuit is an envelope detector which consists of an RF transformer, RF diode, and low pass filter. This AC-coupled modulation technique has been extremely successful in the laboratory. The optical processor has operated for hundreds of hours without any problems from thermal drift at P_3.

4. FINITE ELEMENT LINEAR DYNAMIC CASE STUDY

The case study is a linear dynamic finite element structural mechanics problem. The effects of simulated earthquake loadings applied to a plane frame structure are analyzed by our optical laboratory system. The case study has been fully detailed elsewhere [6], and thus we present only the significant equations which are implemented on the optical linear algebra processor.

Dynamic analysis [17] of the 11-node structure chosen requires solution of the matrix equation

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{p}(t), \tag{1}$$

where M, C, and K are the 33 × 33 mass, damping, and stiffness matrices, the vector $\mathbf{p}(t)$ is a vector of any externally applied time-varying loads, and $\ddot{\mathbf{d}}, \dot{\mathbf{d}}$ and $\mathbf{d}$ are the acceleration, velocity, and displacement vectors, respectively. The matrices M, C, and K describe the distribution of mass, damping, and elasticity throughout the structure, respectively. We consider a linear analysis, i.e. the mass, damping, and stiffness matrices remain constant throughout the problem solution. Equation 1 is a system of second order linear differential equations that must be solved for the unknown vectors $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and $\mathbf{d}$.

Equation 1 is partitioned into two equations, the first including the nodes where $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and $\mathbf{d}$ are unknown, and the second including the nodes where these values are specified by the earthquake ground motion. These equations are rearranged and solved for the unknown values of $\ddot{\mathbf{d}}$, $\dot{\mathbf{d}}$, and $\mathbf{d}$ using the Newmark direct integration algorithm [17]. The most computationally burdensome step [6] of the solution processes is the evaluation of

$$\hat{\mathbf{p}}_{t+\Delta t} = \mathbf{p}_{t+\Delta t} + \mathbf{M}(a_0\mathbf{d}_t + a_2\dot{\mathbf{d}}_t + a_3\ddot{\mathbf{d}}_t) + \mathbf{C}(a_1\mathbf{d}_t + a_4\dot{\mathbf{d}}_t + a_5\ddot{\mathbf{d}}_t), \tag{2}$$

where the a_n are constants. The evaluation of (2) requires calculation of the two banded matrix-vector products involving M and C, as well as three additional banded matrix-vector products needed to compute $\mathbf{p}_{t+\Delta t}$. We solve equation (2) on the optical linear algebra processor in both the laboratory and the simulator.

The Newmark direct integration algorithm requires 600 iterations for our case study, which requires equation (2) to be solved at 600 time steps. We consider the horizontal displacement of a significant node (the top left node) of the structure at all 600 iterations for the case study solution output. The solution is shown in Figure 3. The horizontal displacement in inches is plotted on the vertical axis vs. the 600 time steps of the 10 second tremor on the horizontal axis. The solution follows the ground motion displacement superimposed with an additional 3.67 Hz oscillation, characteristic of the fundamental oscillatory mode of the structure.

5. ERROR SOURCES, MODELING, AND SIMULATIONS

Previous work in error source modeling for optical numerical processors has focused on error source analysis of a frequency-multiplexed AO systolic space-integrating processor operating on analog data [18]. Models verified in the laboratory have shown that the individual errors combine in a mean square sense for the analog processor [19]. Error source modeling for the present digitally encoded processor was performed [5], and showed that the modeling for digitally encoded optical processors requires extensive computer time. The error modeling (this section) uses the Cray X-MP/48 and significant simplifications to reduce the modeling complexity and computational requirements. These simplified models are verified with laboratory data in Section 6.

Our Sharp laser diodes are weakly index-guided laser diodes and are similar in operating characteristics to channeled substrate planar laser diodes [20]. There are four significant noise sources associated with laser diodes: intrinsic intensity noise (quantum noise); mode partition noise; optical feedback noise; operating curve kinks and self-pulsations. Intensity noise is the only noise source of concern for our processor. Since we detect the total optical intensity and have no chromatic dispersion mechanism in our system, mode partition noise does not exist. Optical feedback noise from near-end and far-end reflections [21] are negligible in our system, due to the detection bandpass frequencies and the system

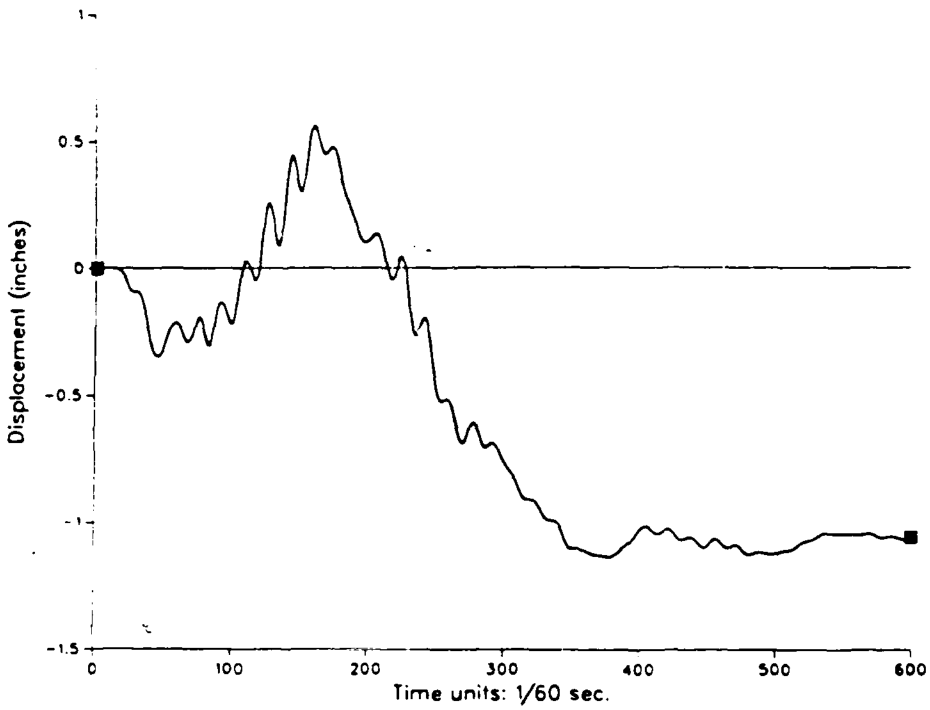


Figure 3: Case study solution.

design. There are also no operating curve kinks at our levels of operation, or any self-pulsations associated with our devices. *Intensity noise* fluctuations are due to the response of the optical field in the laser diode to the modulation by the intrinsic shot noise of the carriers. For a fixed bias level, an increase in the modulation depth produces an increase in the intensity noise, thus intensity noise is signal dependent (multiplicative) and time-varying for our modulation scheme. For our laser diodes, the spectrum of the intensity noise is nearly flat up to 1 GHz. The time-varying intensity variations may be modeled by a Gaussian density [22].

Another P_1 error source is due to the difference between the transfer functions of the P_1 point modulators in the input array. This causes a spatial error across the input array which we refer to as input *spatial gain error*. This error includes effects from the individual D/A converters and RF mixers feeding the laser diodes, as well as the laser diodes themselves. The error is fixed and multiplicative, and is considered to be the residual error after calibration. Residual errors are properly modeled by a truncated Gaussian density, where the maximum variation is limited to 3σ by the calibration and measurement accuracy.

The significant P_2 AO cell errors are acoustic attenuation and crosstalk. Acoustic attenuation is correctable for a single frequency by using a fixed mask before or after P_2, or by adjusting the gains of the P_1 point modulators. For a narrowband modulation signal such as ours, these methods permit correction to within a small fixed multiplicative residual error. Multi-channel AO cells can be designed to give acoustical and electrical crosstalk isolation levels of 30 dB or better. This is the isolation obtained with our P_2 multi-channel AO cell. This error source is not explicitly included for reasons discussed later.

Each P_3 detector system channel includes a silicon p-i-n detector, its wideband amplifier, demodulator, and A/D converter. Quantum noise, or *shot noise*, is present in the photocurrent due to the quantized nature of charge carriers. We are interested in the mean square noise current because it is a measure of the noise variance, and hence the standard deviation, which is used in our error modeling. The total mean square shot noise current i_s is given by

$$\overline{i_s^2} = 2eBI_t, \tag{3}$$

where B is the detection system bandwidth, e is the electronic charge, and I_t is the total photocurrent. I_t consists of signal, dark current, and background radiation components, where the latter two are insignificant in our system. Shot noise is time-varying and signal dependent (multiplicative) with a white frequency spectrum [23]. Shot noise is governed by a Poisson density which may be closely approximated by a Gaussian density [24].

A thermal (Johnson, Nyquist) noise current i_T arises in the load resistor R_L of the photodiode. It is temperature dependent and has a mean square value of

$$\overline{i_T^2} = \frac{4KTB}{R_L}, \tag{4}$$

where K is Boltzmann's constant, and T is the absolute temperature. Thermal noise has a white spectrum and is also governed by a Gaussian density [24]. It is time-varying and signal independent (additive). The detector amplifier contributes additional noise to the signal from resistive elements and active components in the amplifier. This noise current has both thermal and shot noise components, but the thermal noise dominates, and thus the amplifier noise is primarily signal independent (additive). We refer to the thermal noise in R_L and the detector amplifier as *detector noise*. The demodulator circuits and A/D converters also contribute noise, which we refer to as *circuit noise*. We use a first-order approximation to random noise in the A/D converter circuit by modeling a perfect A/D converter preceded by signal independent circuit noise. The demodulator circuit noise is due to thermal noise in the components being used. We thus model both the detector and circuit noise as time-varying and additive. The N channel P_3 detector system array also exhibits spatial gain error due to the variations in detector channel transfer functions. We refer to this as *spatial response error*, which is a fixed multiplicative residual error.

We now present the specific error models we use to simulate the optical processor. Several simplifications are made to reduce the complexity and computational requirements of the models. The error sources may be classified as multiplicative or additive, and fixed or time-varying. For a multiplicative error, the noise or error value multiplies the desired signal, and the total noise contribution is thus signal dependent. For an additive error, the noise value is added to the signal, and thus the total noise contribution is independent of the signal. Fixed errors are constant, while time-varying errors are different for each T_1 time interval.

We model two P_1 errors, fixed *spatial gain* error and time-varying *intensity noise*. Both error sources are signal dependent (multiplicative) in nature. No signal independent P_1 error is included. Four P_3 detector system error sources are modeled. The first two are multiplicative. They include the fixed *spatial response* error (which models the residual detector channel response and non-uniformity errors), and a time-varying *shot noise* error (which models the photodiode shot noise). The remaining two P_3 error source models are additive and time-varying. They include *detector noise* (modeling the thermal noise from the load resistor R_L and the detector amplifier noise), and *circuit noise* (which includes the demodulator and A/D circuits).

The error models and their notation are summarized in Table 1, which notes the specific error source, the error type, and the model used for it. We denote the M input P_1 point modulators by the subscript i (which can take on the values 1,...,M), and the N detector P_3 channels by the subscript j (which can take on the values 1,...,N). The superscripts 1 and 3 denote a P_1 error and a P_3 error respectively.

plane	model	type	notation
P_1	spatial gain	fixed multiplicative	$1+\delta_i^1$
	intensity noise	time-varying multiplicative	$1+\theta(t)$
	spatial response	fixed multiplicative	$1+\delta_j^3$
P_3	shot noise	time-varying multiplicative	$1+\theta(t)$
	detector noise	time-varying additive	$d_j(t)$
	circuit noise	time-varying additive	$c_j(t)$

Table 1: Error models and notation.

To formulate the fixed multiplicative spatial gain and intensity noise P_1 error models, we write the exact input to laser diode i as a_i and describe the optical intensity leaving it as

$$\hat{a}_i = a_i(1+\delta_i^1). \tag{5}$$

The optical signal b_j incident on P_3 detector channel j is thus

$$b_j = \hat{a}_i x_{ij}, \tag{6}$$

where $\hat{a}_i$ is given in equation (5) and x_{ij} represents the AO cell transmittance at P_2. The detector channel's output just before the perfect A/D converter is

$$\hat{b}_j = b_j(1+\delta_j^3)(1+\theta(t)) + d_j(t) + c_j(t). \tag{7}$$

The error model simulations are substantially simplified by using the same $\theta(t)$ in both the P_1 and P_3 time-varying multiplicative error models. This approximation has been justified as discussed later.

The noise value δ_i^1 for the spatial gain error is simulated by

$$\delta_i^1 = \sigma D, \tag{8}$$

where D is a random zero-mean Gaussian variate of unit variance N(0,1), and σ is the desired standard deviation of the error. The maximum percent error (MPE) that is expected for a given value (or that we wish to model) determines σ from

$$3 \times \sigma \times \ 100\% = MPE. \tag{9}$$

Since more than 99% of the Gaussian variates are within 3σ, this model accurately describes greater than 99% of the errors. The fixed variates that are used in the simulations are checked to make sure that no $\mid D \mid > 3$ value exists outside of the 3σ range. This prevents any unreasonably large variates, which are not present for fixed errors that are calibrated to within a measured value. The possibility of a variate $\mid D \mid > 3$ for a time-varying error is realistic because of the Gaussian nature of the various noise sources. We use (8) and (9) (with different σ values) to describe all of the multiplicative errors, δ_i^1, δ_j^3 and $\theta(t)$. New Gaussian variates are generated at every T_1 in the processor simulations for the time-varying errors.

The zero-mean Gaussian additive noise values $d_j(t)$ and $c_j(t)$ are simulated similarly to the multiplicative noise values. However, since these errors are additive, reference to the full-scale (FS) value of the signal being modeled is included, i.e.

$$d_j(t) = \sigma D, \tag{10}$$

where σ satisfies

$$3 \times \sigma \times 100\% = MPFSE \times FS, \tag{11}$$

where MPFSE is the maximum percent full-scale error. The circuit noise $c_j(t)$ is also simulated by (10) and (11), with a different σ. The FS value for the circuit noise and detector noise is the maximum possible mixed-radix detector value. For the processor size M and radix B, the full-scale value is

$$FS = M(B-1)^2. \tag{12}$$

The simulator (and the actual A/D used) incorporates FS clipping, whereby any noise-corrupted detector value that is outside of the A/D range is clipped to the maximum or minimum A/D input analog value.

We implement an important modeling simplification by including no P_2 error sources. This is valid because any residual acoustic attenuation error may be included in the P_1 fixed spatial gain error, and the crosstalk effects are insignificant for our encoding with 30 dB of isolation. Crosstalk effects may also be included indirectly in the P_3 multiplicative time-varying error. Another simplification is the consolidation of the additive time-varying P_3 error sources $d_j(t)$ and $c_j(t)$. The error sources are combined by summing the variances calculated from (11). The final simplification is the merging of the P_1 and P_3 time-varying multiplicative intensity noise and shot noise errors into one generated error signal $\theta(t)$. This was done since an extremely time-consuming computational portion of the error source model simulation is the generation of time-varying variates every T_1 [5]. In our simplified model, we generate only a single variate every T_1 which is used for all of the M-channel multiplicative intensity noise errors and all of the N-channel multiplicative shot noise errors. The single variate (independent of i and j) is used for the $\theta(t)$ noise value in Table 1. The simplification is justified if the mean and the variance of $\hat{b}_j$ are the same with and without the simplification. It is simple to show that the means are equivalent. We generate the σ value for the $\theta(t)$ noise value to make the variances equivalent. These error model simplifications significantly reduce the complexity and the required computation time of the optical processor simulator. Even so, the simulation of the implementation of our dynamic case study requires a large amount of computer time. The optical processor simulations were run on a Cray X-MP/48 supercomputer. The supercomputing facilities are part of the Pittsburgh Supercomputing Center, and the computer time was arranged by the National Science Foundation. A radix 2 simulation of our case study requires 600 seconds (10 minutes) of Cray CPU time.

The goals of the error model optical processor simulations are (1) to understand how individual and combined errors affect the optical processor and determine the dominant errors, and (2) to evaluate the error models' ability to properly model the optical processor and predict its performance; this involves verifying the simulation results with laboratory data (Section 6). Errors will occur in the processor whenever a noise source causes a mixed-radix optical signal at P_3 to be incorrect by 1/2-bit or greater before the A/D conversion operation takes place. We refer to this occurrence as an output error. Any output error will cause the case study solution to be in error as well. We refer to this error as solution percent error, which is defined by

$$solution\ percent\ error = \frac{1}{600}\sum_{i=1}^{600}\left[\left|\frac{e(i)-g(i)}{g(i)}\right| \times 100\%\right], \tag{13}$$

where $e(i)$ is the calculated displacement at iteration i for the error simulation, and $g(i)$ is the displacement value with no processor errors.

We expect the additive time-varying noise sources (detector and circuit noise) to dominate in the optical processor. As the MPE level of a multiplicative error is increased, the number of possible multiplications which can result in an output error increases. Thus, in general, we expect the solution percent error to increase proportional to the MPE level for multiplicative errors. However, as an additive error MPFSE is increased, a sharp transition occurs from a condition where no output errors can occur, to one where an output error may occur for every possible multiplication. Thus, we expect a large jump in the solution percent error value from zero, as the MPFSE of an additive error is increased. We also expect multiple error sources tc combine nonlinearly [5] because of the A/D conversion operation. Several error sources, each too small to cause output errors individually, may combine to reach the threshold where output errors occur. Thus, when multiple error sources are combined, the results cannot be predicted by a linear combination of the individual error sources and their separate effects.

Table 2 shows selected simulation results for radix 2 operation. Tests 1-8 indicate the individual error source MPE and MPFSE ranges where solution percent errors first occur. Tests 1,2 and 5,6 show that solution percent errors will occur for MPE values between 55 and 70 for the fixed multiplicative P_1 and P_3 spatial errors. Tests 3 and 4 show that the first solution percent errors occur when the MPE is between 30 and 35 for the time-varying P_1 and P_3 intensity noise and shot noise errors, which are simulated with the same variates and error value $\theta(t)$. Tests 7 and 8 show the results for the time-varying additive P_3 errors. Because these errors are simulated together by adding their variances, the results indicated in tests 7 and 8 are equivalent to either individual error with a MPFSE value of that in test 7 and 8 multiplied by $\sqrt{2}$. Test 9 indicates one set of multiple error levels that can be allowed with no solution errors.

test no.	spatial gain MPE fixed mult.	intensity noise MPE time-var. mult.	spatial response MPE fixed mult.	shot noise MPE time-var. mult.	detector noise MPFSE time-var. additive	circuit noise MPFSE time-var additive	solution percent error
1	60.0	0.0	0.0	0.0	0.0	0.0	0
2	70.0	0.0	0.0	0.0	0.0	0.0	95
3	0.0	30.0	0.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	35.0	0.0	0.0	29
5	0.0	0.0	55.0	0.0	0.0	0.0	0
6	0.0	0.0	65.0	0.0	0.0	0.0	120
7	0.0	0.0	0.0	0.0	19.93	19.93	0
8	0.0	0.0	0.0	0.0	19.94	19.94	75×10^{3}
9	3.0	6.0	3.0	6.0	10.0	10.0	0
10	10.0	7.6	10.0	7.6	14.4	14.4	0
11	10.0	7.6	10.0	7.6	19.2	19.2	59×10^{-4}

Table 2: Radix 2 simulation results.

The simulation results for all radices clearly indicate that the additive time-varying P_3 errors are the dominant error sources. The detector and circuit noise MPFSE values required to cause a non-zero percent error in the solution are lower than the MPE values for any other error source. The jump in percent error as the MPFSE is increased over that level (tests 7 and 8) is considerably more drastic than the jump in percent error for the other processor error sources (tests 1-6), as expected. The last three tests listed in Table 2 show simulation results for all error sources combined. These show comparisons between different error source MPE and MPFSE levels which yield little or no solution percent error. Tests 10 and 11 show the nonlinear thresholding effect predicted for combined error sources, with the additive time-varying MPFSE noise levels increased in test 11 from their value in test 10. The solution percent error is small in test 11 (since other MPE errors are present), and not the large jump that is observed when only the additive MPFSE errors are present (tests 7 and 8).

6. LABORATORY RESULTS

This section presents laboratory results which verify the error source modeling and simulations for the optical processor. The various MPE and MPFSE error source levels were measured in the laboratory for our M=N=1 channel processor. The measurements were made from an RF spectrum analyzer or an oscilloscope. The typical error source levels for our laboratory optical processor are given in test 1 of Table 3. The 3.00 MPFSE for the detector noise is the most troublesome because it is the largest error source value, and detector noise is a dominant error in the processor. This large value was primarily due to the low input impedance design of our detector amplifier which contributes a great deal of thermal noise current at the input (equation (4)). The last two columns of Table 3 for test 1 show that the case study runs with 0.0% solution percent error on the laboratory processor, as well as on the simulator.

test	spatial gain MPE fixed mult.	intensity noise MPE time-var. mult.	spatial response MPE fixed mult.	shot noise MPE time-var. mult.	detector noise MPFSE time-var. additive	circuit noise MPFSE time-var. additive	laboratory percent error	simulation percent error
1	1.00	1.44	1.00	0.02	3.00	0.42	0.00	0.00
2	1.00	1.44	60.00	0.02	10.0	0.42	0.70	0.66
3	30.0	25.0	1.00	0.02	3.00	0.42	25.6	24.1
4	1.00	1.44	1.00	0.02	20.00	20.00	1226	7490

Table 3: Error model performance evaluation results.

Tests 2-4 of Table 3 verify the simulator's ability to accurately predict the performance of the laboratory optical processor. In each test, various errors are artificially introduced into the laboratory processor setup, over the nominal values given in test 1. In test 2, the spatial response MPE is increased to 60.00, and the detector noise is increased to 10.00. The solution percent error obtained by running the case study on the laboratory optical processor was 0.70%, as shown in test 2 of Table 3 and in Figure 4, where the solid line represents the exact (no error) optical processor solution, and the dashed line represents the laboratory results. The laboratory results are quite close to the exact results, with small errors visible near iterations 280 and 360. To test our error modeling and simulator, a simulation was run using the error model values in test 2. The simulator case study solution is shown in Figure 5. The percent error predicted by our simulator was 0.66 (test 1), which differs by 0.04% and is within 6 percent ($\frac{0.04 \times 100\%}{0.70}$ = 6%) of the value (Figure 4) determined on the optical laboratory system. The general shape of both curves are nearly identical, with no large visible error oscillations along the solution path. Thus, the simulator accurately predicted the laboratory processor's performance for this set of error model levels.

Test 3 in Table 3 used an effective spatial gain MPE of 30.0 (30 times the normal system value), and an effective intensity noise MPE of 25.0 (17 times the normal system value). The optical laboratory

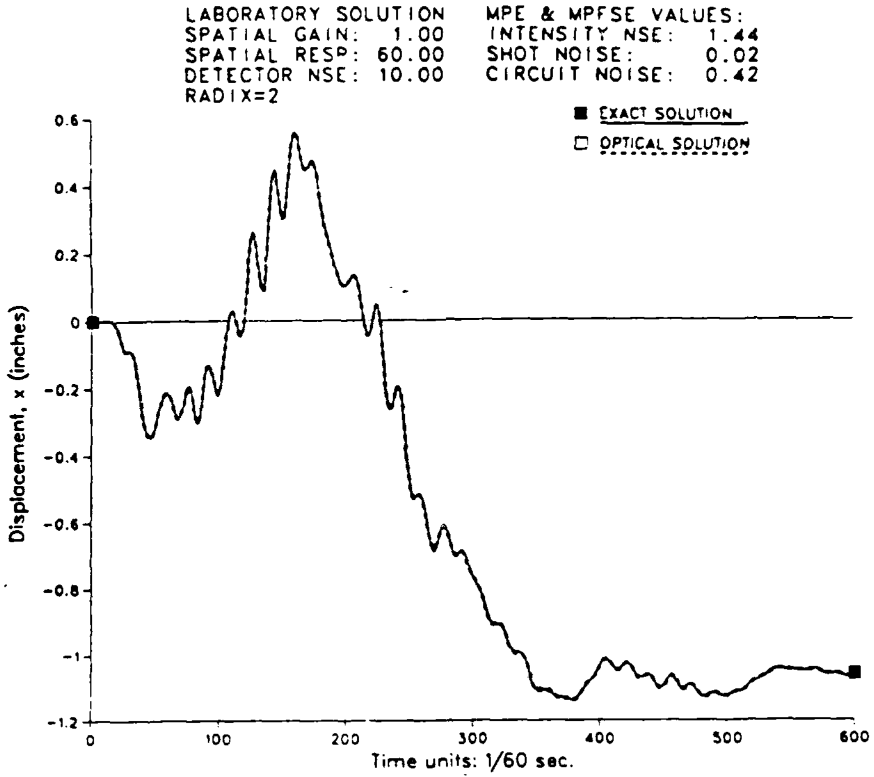


Figure 4: Laboratory case study with 0.70% solution error.

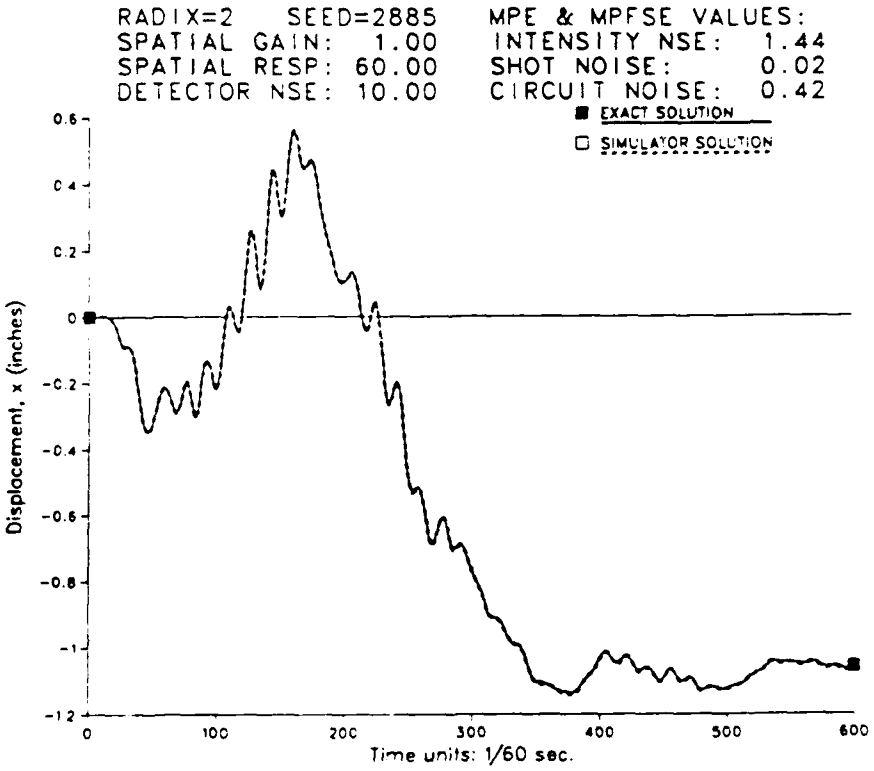


Figure 5: Simulated case study with 0.66% solution error.

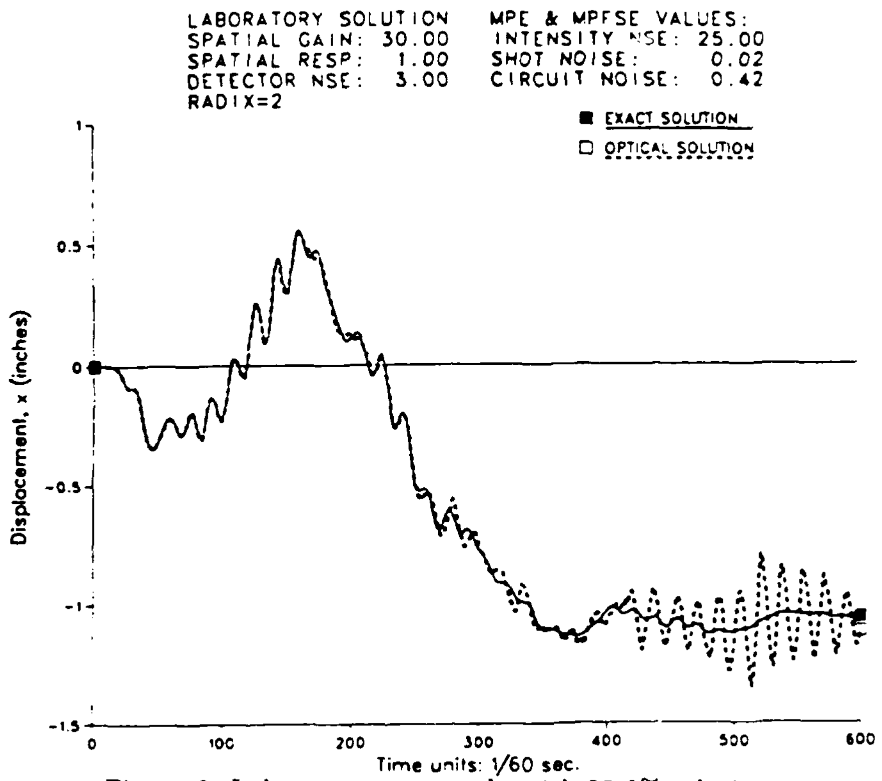


Figure 6: Laboratory case study with 25.6% solution error.

case study output is shown in Figure 6. The figure shows a significant solution percent error which is calculated to be 25.6%. The output of the simulator run using these error model levels (test 2) is shown in Figure 7. As seen, the simulated solution is similar to the laboratory solution shown in Figure 6, with both solutions having moderate error oscillations for approximately 250 iterations. The simulated percent error is 24.1%, or a 1.5% difference, which is an accurate difference of $\frac{1.5 \times 100\%}{25.6} = 6\%$ in the laboratory processor's performance versus that of the simulator.

Test 4 in Table 3 used the laboratory processor with large effective detector and circuit noise MPFSEs of 20 (7 and 48 times the typical system values). The laboratory solution results in Figure 8 show a fixed-point representation clipping phenomenon, due to nonlinear clipping effects of the limited dynamic range 48-bit binary encoding, which clips the response at an amplitude near 1.0. The high frequency response is due to excitation of higher structural oscillatory modes, nonlinear effects, or both. The laboratory result has a large percent error of 1226%. The simulated result with the MPFSE levels in test 4 (Table 3) is shown in Figure 9, where clipping is exhibited as well. The simulator percent error was 7490%. We expect this to be large, but do not expect it to necessarily agree well with the laboratory value when such large processor errors are involved. This is due to the dependence of additive error on the specific simulator fixed variate values, and the nonlinear nature of the fixed-point clipping effect. It is important that the clipping effect was predicted by the simulator, and that a similar invalid case study solution was obtained.

In the above three independent trials with different values for four error sources, our simulator accurately predicted the laboratory case study solution. The basic variations in the shape of the curves for the two solutions (laboratory and simulator) are similar, and the percent error values are in excellent

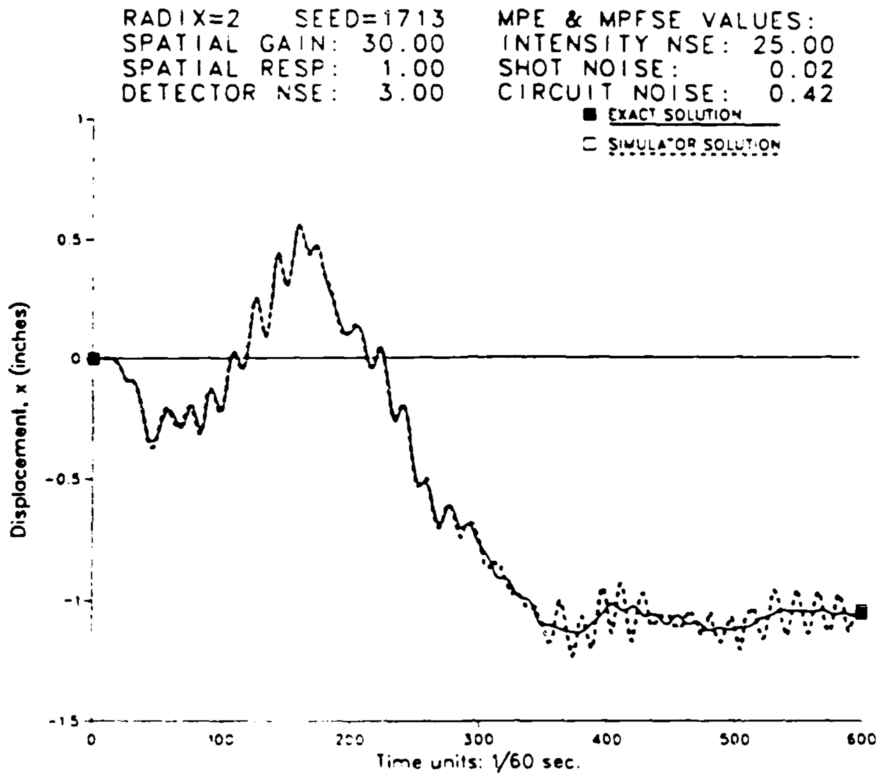


Figure 7: Simulated case study with 24.1% solution error.

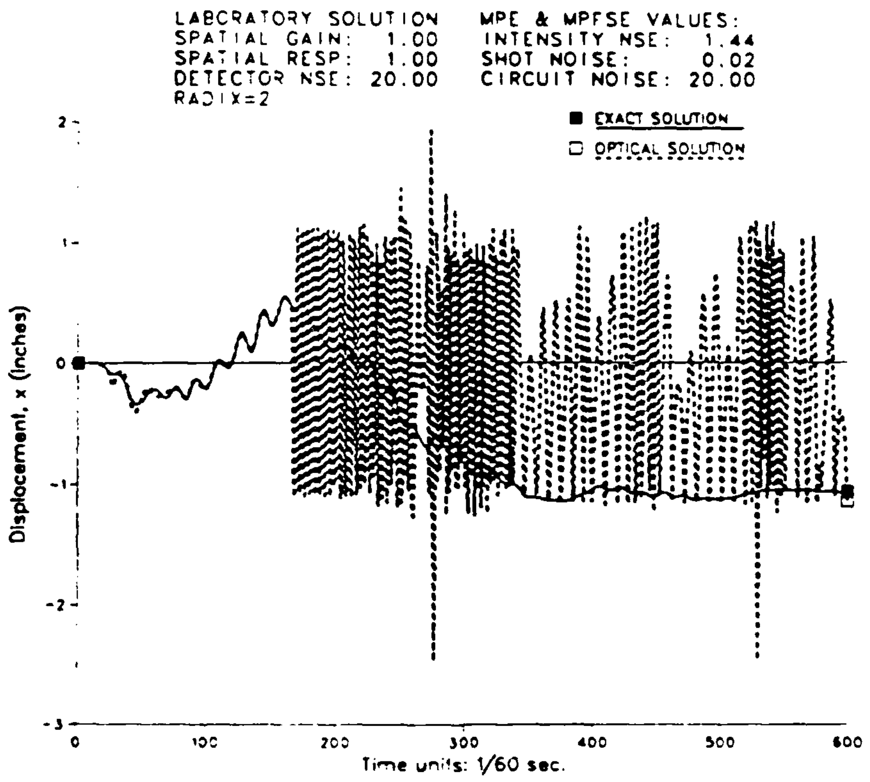


Figure 8: Laboratory case study with 1226% solution error.

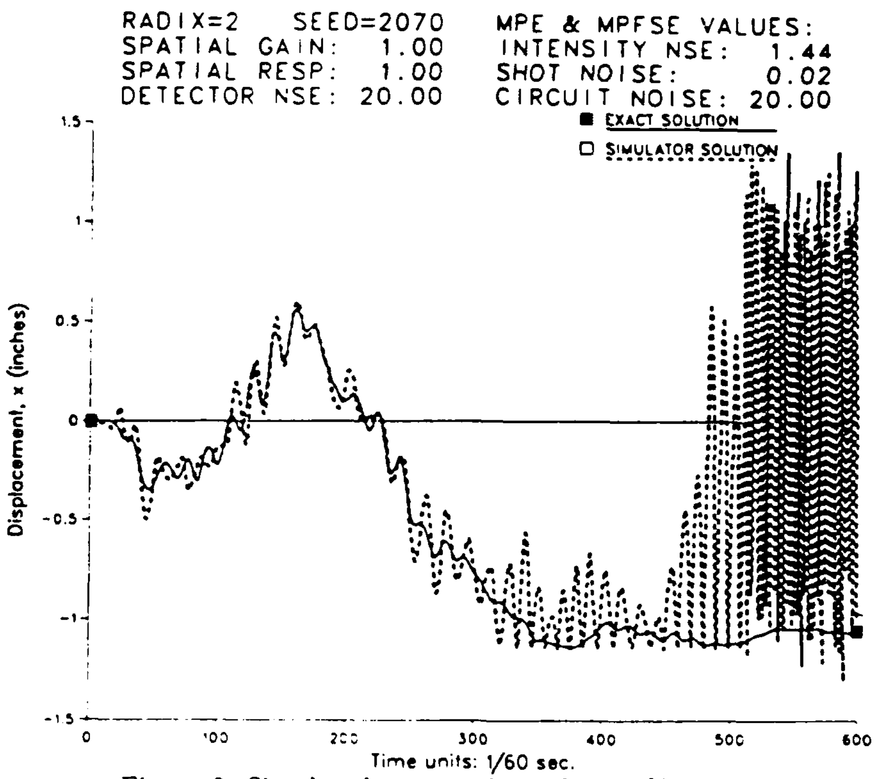


Figure 9: Simulated case study with 7490% solution error.

agreement (within 6 percent error for small and moderate percent error values, and of the same order of magnitude for large error values).

We now use the simulator to evaluate the potential of the laboratory optical processor. Our laboratory processor was built as a proof-of-principle system using new P_1 point modulators and a sub-optimally designed detector amplifier. The laboratory electronics, RF circuitry, and support hardware were designed to be general and flexible enough to serve a variety of laboratory needs. These components were obtained with a moderate budget, and have worked well for our research purposes. However, these components do not give the optimal performance that could be obtained with custom designed hardware. For example, if a properly designed transimpedance detector amplifier is used at P_3, the detector noise MPFSE can be reduced by several orders of magnitude. We summarize the best possible MPE and MPFSE values possible with current technology for the error sources values in our processor in Table 4.

spatial gain MPE fixed mult.	intensity noise MPE time-var. mult.	spatial response MPE fixed mult.	shot noise MPE time-var. mult.	detector noise MPFSE time-var. additive	circuit noise MPFSE time-var. additive
0.01	0.04	0.01	0.02	0.002	0.01

Table 4: Achievable optical processor error source levels.

We evaluate the optical processor's performance potential by considering the maximum number of input channels M possible with radix 2, 4, and 8 encoding for the error source values given in Table 4.

These maximum M values are listed in Table 5. Test 1 of Table 5 shows that the optical processor can utilize more than M=1000 channels for radix 2 encoding. Practical considerations limit the number of input channels M to approximately 200. Test 2 shows that a maximum of M=150 channels may be used for radix 4 encoding. The number of input channels is substantially limited for radix 8 encoding. Test 3 shows that a maximum of 23 channels may be used. This is because we require $M(B-1)^2 = 49M$ resolvable levels at P_3, plus a maximum of 3σ error variation to ensure no digital processing errors. These restrictions approach the analog dynamic range limitations of the system, as specified in Table 4.

test no.	radix	Maximum channels M for zero percent error
1	2	> 1000
2	4	150
3	8	23

Table 5: Maximum simulated M yielding zero percent error, using Table 6.4 error source levels.

Table 6 presents several optical linear algebra processor specifications using the maximum M given in Table 5 (with $M = 200$ for radix 2). The processor performance is measured in millions of operations per second (MOPS), where one operation is defined as one multiplication and one addition. The table shows that for binary encoding and M=200 channels, 32-bit multiplications may be performed at a rate of 625 MOPS with a processor using only 8-bit A/Ds.

radix	N	M	A/D bits	T_2 ($nsec$)	MOPS
2	32	200	8	320	625
4	32	150	11	320	469
8	32	23	11	320	72
2	16	200	8	160	1250
4	16	150	11	160	938
8	16	23	11	160	144

Table 6: Various optical processor performance measures for 32-bit and 16-bit multiplications with 100 MHz 12-bit A/Ds.

7. SUMMARY

This paper has presented measurements and performance results obtained from the laboratory optical processor. Noise measurements for the six error model components were detailed for our laboratory optical processor. The error modeling was shown to be accurate, as was verified by four different laboratory processor case study solution results with increased errors. Both the percent error values and the solution trajectory form were predicted properly by the error source simulator. We discussed the error source levels that could be achieved if the most advanced technology was used to fabricate the optical processor. These levels were quantified for an $M > 1$ channel processor to determine its maximum performance capability. We found that 200 channels may be used with radix 2 encoding, 150 channels with radix 4 encoding, and 23 channels with radix 8 encoding, and that performance above 1 GOPs is possible.

The simplified error models are not limited to the area of linear algebra processors. These contributions are applicable to a wide variety of optical systems, including many digital optical computing systems, optical switching and connection systems, optical neural network architectures, AO-based synthetic aperture radar processors, optical correlation systems, and many other general optical computing architectures.

References

[1] "Special issue on optical computing,". *Proceedings IEEE*, Vol. 72, No. 7, July, 1984.

[2] D.P.Casasent, "Acoustooptic linear algebra processors: Architectures, algorithms, and applications," *Proc. of IEEE*, vol. 72, pp. 831–849, July 1984.

[3] D.P.Casasent and B.K.Taylor, "Banded-matrix high-performance algorithm and architecture," *Applied Optics*, vol. 24, pp. 1476–1480, 15 May 1985.

[4] D. Casasent and B. K. Taylor, "Optical finite element processor," *Proc. SPIE*, vol. 564, pp. 139–149, August 1985.

[5] B.K.Taylor and D.P.Casasent, "Error-source effects in a high-accuracy optical finite-element processor," *Applied Optics*, vol. 25, pp. 966–975, 15 March 1986.

[6] B. Taylor and D. Casasent, "Optical matrix-vector laboratory data for finite element problems," in *Proc. SPIE*, vol. 939, April 1988.

[7] D. Casasent and S. Riedl, "Time and space integrating optical laboratory matrix-vector array processor," *Proc. SPIE*, vol. 698, pp. 151–156, August 1986.

[8] E. Swartzlander, "The quasi-serial multiplier," *IEEE Trans. on Comp.*, vol. C-22, pp. 317–321, April 1973.

[9] H. Whitehouse and J. Speiser, "Linear signal processing architectures," in G. Tacconi, ed., (*Aspects of Signal Processing - Part II*), (Boston, MA), NATO Advanced Study Institute, 1976, pp. 669–702.

[10] D. Psaltis, D. Casasent, D. Neft, and M. Carlotto, "Accurate numerical computation by optical convolution," *Proc. SPIE*, vol. 232, pp. 151–156, April 1980.

[11] J. Goodman, et al. "Parallel incoherent optical vector-matrix multiplier,". BMD Technical Report No. L-723-1, February 1979.

[12] D.P.Casasent, J.Jackson, and C.P.Neuman, "Frequency-multiplexed and pipelined iterative optical systolic array processors," *Applied Optics*, vol. 22, pp. 115–124, 1 January 1983.

[13] R.P. Bocker, S.R. Clayton, and K. Bromley, "Electrooptical matrix multiplication using the twos complement arithmetic for improved accuracy," *Applied Optics*, vol. 23, pp. 2019–2021, July 1983.

[14] B.K.Taylor and D.P.Casasent, "Twos-complement data processing for improved encoded matrix-vector processors," *Applied Optics*, vol. 25, pp. 956–961, 15 March 1986.

[15] C.Perlee and D.P.Casasent, "Negative base encoding in optical linear algebra processors," *Applied Optics*, vol. 25, pp. 168–169, 15 January 1986.

[16] Sharp Electronics, Japan, *Laser Diode User's Manual: Sharp Laser Diodes*, 1985.

[17] K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis.* Englewood Cliffs, NJ.: Prentice-Hall, Inc., 1976.

[18] D.P.Casasent and A.Ghosh, "Optical linear algebra processors: Noise and error source modeling," *Optics Letters*, vol. 10, pp. 252–254, June 1985.

[19] D. Casasent and J. Jackson, "Space and frequency-multiplexed optical linear algebra processors: Fabrication and initial tests," *Applied Optics*, vol. 25, pp. 2258–2263, 15 July 1986.

[20] T. Hayakawa, N. Miyauchi, S. Yamaoto, S. Yano, and T. Hijikata, "Highly reliable and mode-stabilized operation in v-channeled inner strip lasers on p-GaAs substrate emitting in the visible wavelength region," *Journal of Applied Physics*, vol. 53, pp. 7224–7234, November 1982.

[21] O. Hirota, Y. Suematsu, and K. Kwok, "Properties of intensity noises of laser diodes due to reflected waves from single-mode optical fibers and its reduction," *IEEE Journal of Quantum Electronics*, vol. QE-17, pp. 1014–1020, June 1981.

[22] W. Tsang, ed., *Lightwave Communications Technology*, Vol. 22 of *Semiconductors and Semimetals*, Part B: Semiconductor Injection Lasers, I. Orlando: Academic Press, 1985.

[23] T. E. Jenkins, *Optical Sensing Techniques and Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall International, Inc., 1987.

[24] G. Keiser, *Optical Fiber Communications.* New York: McGraw-Hill Book Company, 1983.

CHAPTER 5:

OPTICAL MATRIX-VECTOR PROCESSING FOR COMPUTATIONAL FLUID DYNAMICS

Proc. SPIE
Vol. 936
April 1988

936-32

Optical Matrix-Vector Processing for Computational Fluid Dynamics

Caroline J. Perlee and David P. Casasent
Carnegie Mellon University
Dept. of Electrical and Computer Engineering
Center for Excellence in Optical Data Processing
Pittsburgh, Pennsylvania 15213

ABSTRACT

An optical processor to solve partial differential equations for computational fluid dynamics applications is considered. This application is new and original for optical processors. The algorithms that are used are optical realizations of the Newton-Raphson method for nonlinear equations and a new optical LU direct decomposition and Gauss-Seidel iterative solution to the resultant linear algebraic equations. These algorithms are used to solve Burger's equation (a specific form of the momentum equation in fluid dynamics). The nonlinear equations provide 1-D velocity data at each time step. Simulation results of optical processing with these algorithms on computational fluid dynamics data is included.

1. INTRODUCTION

Optical linear algebraic processors (OLAPs) are general-purpose optical array processor systems that perform matrix-vector multiplications and a wide variety of linear algebraic operations [1]. To obtain high-accuracy, we use the digital multiplication by analog convolution (DMAC) algorithm [2-4] with multi-level (non-base 2) encoding [5] and a negative base [6] to handle bipolar data. This paper addresses the issues of accuracy, speed, performance and the base used for two linear algebraic equation (LAE) solution algorithms to solve nonlinear, time-varying partial differential equations (PDEs).

We use a 1-D PDE example from computational fluid dynamics (CFD) as our case study. This is a new application for optical matrix-vector processing and is an opportunity to analyze and quantify OLAP performance in the complex computational requirements of fluid dynamics. Our case study is the momentum equation from the Navier-Stokes equations without the pressure term; it is also known as Burger's equation. It is a PDE that is nonlinear in velocity and time-varying. This problem concerns a 2-D box containing a viscous fluid. Two waves exist in the fluid with different velocities and traveling in different directions (positive and negative velocities). These fluid waves travel in time, cross and eventually damp to zero. We consider the 1-D version of this problem to demonstrate the point. We are concerned with the spatial distribution of velocities in time. The problem is formulated using the finite element method to approximate the velocity function in space (as the sum of basis functions composed of two triangular shape functions per element) and finite difference methods to approximate the time derivatives [7]. We use finite elements since they are generally more accurate (for the same number of calculations) than are finite differences in approximating the spatial derivatives (i.e., they achieve the same accuracy as finite difference techniques but with a coarser grid and therefore require fewer calculations). These methods discretize the partial differential equations and generate a set of time-dependent, nonlinear algebraic equations which are solved by the OLAP. We linearize the nonlinear equations by the Newton-Raphson method, which generates a set of LAEs that we solve by iterative and direct methods. We discuss the processor's performance using the Gauss-Seidel iterative algorithm

936-32

and a new optical implementation of the LU direct decomposition [5] (Section 4). Our tests used three different radices (-2, -4 and -8).

A brief review of the OLAP architecture considered is given in Section 2. The case study problem is detailed in Section 3 together with the algorithms for solving the nonlinear and linear algebraic equations. In Section 4, we tabulate and discuss the results of our simulations. Our conclusions are given in Section 5.

2. Architecture Review

Figure 1 shows the OLAP architecture considered. It is a schematic of a prototype optical linear algebraic processor [5] that has been fabricated [8] in our laboratories. It is a new space- and time-integrating architecture that computes one vector-inner-product (VIP) in a single time step T_2. It allows high-accuracy processing using data encoded in either binary, multi-level or negative bases. Previous research [9,10] has demonstrated its ability to solve a finite element problem in structural mechanics. This architecture can readily accomodate large problems by a unique diagonal partitioning of the matrix [5] and achieve higher accuracy by bit partitioning [8]. The range of applications of the optical processor is quite interdisciplinary since LAE problems arise in many scientific fields.

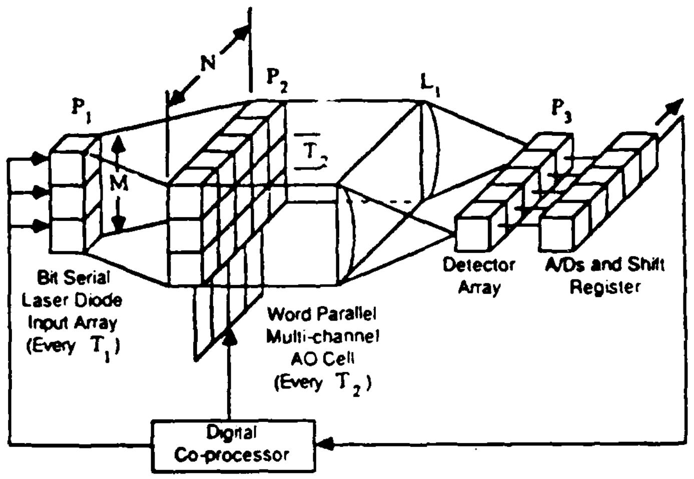

N
P2
L1
P1
P3
M
T2
Bit Serial Laser Diode Input Array (Every T1)
Word Parallel Multi-channel AO Cell (Every T2)
Detector Array
A/Ds and Shift Register
Digital Co-processor

Figure 1: Multichannel high-accuracy optical linear algebra processor

We now briefly describe the operation of the OLAP in Fig. 1. The architecture was introduced several years ago, thus our description and detail of it is brief. The system consists of multiple point modulators at plane P_1, a multiple channel acousto-optic (AO) cell at plane P_2 and a detector array in plane P_3. The plane P_3 detector array consists of a multiple shift/add architecture.

Figure 1 shows M vertical processing channels in P_1 and N horizontal channels in the AO cell in P_2. The operands are N-bit digitally encoded words, i.e. the same number of bits as the number of horizontal channels in the AO cell in plane P_2. Consider one of the M channels in P_1 used to multiply two numbers $Z = XY$. The bits of X are fed bit serially into one point modulator at P_1 and the bits of Y are fed word parallel into P_2. The data in P_2 travels vertically up the AO cell in word parallel form and remains in the region of P_2 opposite one P_1 modulator for a time T_2. Meanwhile, the P_1 point

936-32

modulator is pulsed on N times during T_2, i.e. P_1 data is input at a rate once every time T_1 where $NT_1 = T_2$. Thus, in a time T_2, the N digits of X have been fed to P_1. The light from the P_1 point modulator is imaged horizontally across the region of P_2 containing Y. The N AO cell channels are focused onto the N detector elements in P_3. This N-bit word incident on P_3 is a partial product of XY. The detection system shifts this result and the next partial product is formed and added to it, thus accumulating partial products at P_3. One new digit of Z is output from P_3 every T_1. After $NT_1 = T_2$, the P_3 output is a mixed-radix representation of the product of the two encoded numbers X and Y. This is the DMAC algorithm and it can be applied to data encoded in any radix. Figure 1 shows that the mixed-radix values are A/D converted, sent to the digital co-processor where they are multiplied by the appropriate power of the operating radix r and added to the accumulated P_3 output at each T_1. In this manner, a radix-r encoding of Z is generated after $T_2 = NT_1$ seconds.

In multi-channel operation, M point modulators at P_1 are presented with data. The light from each is imaged across a different vertical region of P_2 containing one set of N-bit operands. The result from P_3, after T_2 seconds, is a VIP of the data at P_1 and P_2, accurate to N bits. The fact that the AO cell data travels up the cell makes vector operations quite efficient, since data can be re-used as it travels along the cell. In matrix-vector operations, the vector is input to P_2 and the matrix to P_1, one row to each point modulator in parallel.

The implementation of a finite element structural mechanics case study [8] demonstrated some of the limitations of the original prototype and led to the design of a more stable system employing a new AC-coupled mode of operation (which is detailed elsewhere [10]) but has the same basic architecture.

2.1 Number Representation and Partitioning

As stated in Section 2, the DMAC algorithm can be used in any base. With higher bases, fewer digits and OLAP channels N are necessary and operation time T_2 decreases, but A/D dynamic range requirements increase. We will analyze these tradeoffs.

To handle large matrix-vector problems (i.e., when the length of the vector exceeds M), we use diagonal partitioning [5] of the matrix. This results in an ordered data flow in which we break the matrix into diagonal ribbons with M diagonals processed at one time. This method is preferable to dividing the problem into smaller blocks which is inefficient due to the large amount of bookkeeping required and the dead time incurred while the P_2 cell is emptied and reloaded with new data.

Since there are no carries in DMAC until the final conversion from mixed-radix to the operating radix, we can process N_L digits of the data, store the results, process the next N_L digits, etc. [8]. This method is known as bit-partitioning. In this way we can achieve any desired accuracy with only N_L channels at P_2 and P_3.

3. Case Study Description

We now detail the case study and algorithms used in our optical processor simulations.

3.1 Burger's Equation

We first detail the CFD equation and the finite element and finite difference discretizations of the

936-32

PDE. Burger's equation with the boundary and initial conditions that we used is

$$\left.\begin{array}{ll}\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, & 0 < x < 1 \\ u(x,0) = u_0(x) & \\ u(0,t) = u(1,t) = 0 & \end{array}\right\} \quad (1)$$

The unknown velocity is u, the fluid viscosity is ν, the spatial and time variables are x and t respectively, and the boundary conditions at $x = 0$ and $x = 1$ are zero. The initial conditions at $t = 0$ are detailed in Section 3.2. We discretize x into twenty finite element mesh nodes between $x = 0$ and $x = 1$ with spacing $h = 1/20$. Using linear finite element functions for the velocity shape functions and constants for the spatial derivatives, Eq. (1) becomes

$$A\dot{u} + B(u)u + Cu = 0, \quad (2)$$

where uppercase letters denote matrices and lowercase letters denote vectors. Because linear approximating functions are used in this 1-D problem, the matrices $A, B(u)$ and C are tridiagonal. Their size is pxp=20x20 since twenty nodes were used in the 1-D finite element mesh. The matrix $B(u)$ is a function of the unknown velocity, u, and thus the term $B(u)u$ is nonlinear in u.

We approximate the time-derivative term in Eqs. (1) and (2) with a central finite difference formula. This approximation provides greater stability in the solution than does the simpler forward difference approximation. Applying this to Eq. (2), the final form of the discretized Burger's equation is

$$[A + \frac{1}{2}\Delta t(B(u^{n+1}) + C)]u^{n+1} = [A - \frac{1}{2}\Delta t(B(u^n) + C)]u^n. \quad (3)$$

The superscripts n and $n + 1$ denote the time index resulting from the central difference formula. With our choice for the spatial step size ($h = 1/20$), we find [7] that the time step Δt=0.005 assures a maximum error between the finite element method and the exact solution of less than 1%. This is compatable with typical CFD goals to determine the velocity distribution in space and time accurate to 1%.

Our problem is thus to solve (3) for the velocity $u(x)$ at different time steps. We write Eq. (3) as

$$D(u^{n+1})u^{n+1} = E(u^n)u^n, \quad (4)$$

where D and E are the matrices in brackets in (3). They are functions of u^{n+1} and u^n (the spatial velocity distribution at time steps $n + 1$ and n). Given the present $u(x)$ at $t = n$ we cannot easily solve (4) for $u(x)$ at $t = n + 1$, since D is a function of this solution. We use the Newton-Raphson algorithm to reduce (4) to the solution of different LAEs at different Newton-Raphson iterations. In the Newton-Raphson algorithm, we first evaluate D using the present u^n solution, i.e. we form

$$D(u^n)u^{n+1} = E(u^n)u^n = e^n, \quad (5)$$

where e^n is a known vector. We then solve (5) for u^{n+1}. We denote this solution as $u^{n+1,r}$, where r is the Newton-Raphson iteration index. We then refine this $u^{n+1,r}$ solution by solving

$$J^{n+1,r}(u^{n+1,r+1} - u^{n+1,r}) = -R^{n+1,r} \quad (6)$$

for a refined solution $u^{n+1,r+1}$. In (6), the matrices $J^{n+1,r}$ (Jacobian) and $R^{n+1,r}$ (residual in Eq. (4)) are

$$J^{n+1,r} = \frac{\partial R^{n+1,r}}{\partial u^{n+1,r}} \quad (7)$$

936-33

$$R^{n+1,r} = [D(u^{n+1,r})u^{n+1,r} - e^{n,0}] \tag{8}$$

and $e^{n,0} \equiv E(u^{n,0})u^{n,0}$, where $u^{n,0}$ is the converged repsonse from the previous time step. New J and R matrices are then calculated (using the new $u^{n+1,r+1}$ estimate), substituted into (6), and (6) is solved for the new $u^{n+1,r+1}$. This cycle is repeated until convergence, $| u^{n+1,r+1} - u^{n+1,r} | < \epsilon_{NR}$, where ϵ_{NR} is the stopping criterion for the Newton-Raphson iterations. The result is the best solution u^{n+1} to (3) or (4) at the present time step $n+1$ for a given ϵ_{NR}. We then solve (5) for the next u^{n+2} and refine this solution using Newton-Raphson (Eq. (6)).

This process thus involves solving the LAE in (5), then solving the LAE in (6), updating (7) and (8) and resolving the new LAE in (6) for refined solutions until it converges to ϵ_{NR}. The above procedure must be repeated for each time step Δt. In our case, there are twenty spatial values $u(x)$ and $\frac{1}{\Delta t} = \frac{1}{0.005} = 200$ time steps. The Newton-Raphson algorithm has thus reduced a nonlinear problem to a sequence of linear problems (LAE solutions) that are iteratively refined. Table 1 lists the steps in the algorithm. The time index is n and the Newton-Raphson interation index at each time step is r. The initial $u^{0,0}$ vector is used as $u^{n,r}$ in the initial Step 2 and we solve for $u^{n+1,r} = u^{1,0}$ initially. In Steps 3 and 4, we refine this solution until $| u^{n+1,r+1} - u^{n+1,r} | < \epsilon_{NR}$. Then, we set $u^{n+1,r+1}$ equal to u at the present time u^n in Step 2 and solve the new Eq. (5) for u^{n+1} at the next time step (actually $n+2$). The residual vector $R^{n+1,r}$ in Step 3 is

$$R^{n+1,r} = [A + \frac{1}{2}\Delta t(B(u^{n+1,r}) + C)]u^{n+1,r} - [A - \frac{1}{2}\Delta t(B(u^{n,0}) + C)]u^{n,0} \tag{9}$$

which is a function of known matrices and vectors including the converged Newton-Raphson response $u^{n,0}$ from the previous time step. The Jacobian matrix is

$$J^{n+1,r} = \frac{\partial R^{n+1,r}}{\partial u^{n+1}} = A + \frac{1}{2}\Delta t(B(u^{n+1,r}) + B(u^{n+1,r}) + C). \tag{10}$$

For each time step, the algorithm requires the solution of the LAE in Eq. (5) in Step 2 and the repeated solution of the LAE in Eq. (6) in Step 3 r times. We ignore the time to calculate the new J and R (as this requires only matrix-vector multiplications and additions). Solving LAEs is thus the most time-consuming and computationally intense operation. We consider an interative method (Gauss-Seidel) and a direct method (LU decomposition) to solve the LAEs in Steps 2 and 3 on the optical processor of Fig. 1.

3.2 Initial Conditions

We now detail the initial conditions at t=0 used in the case study. It has been shown [7] that the analytical solution to Eq. (1) is

$$u(x,t) = \frac{2\pi(\frac{1}{4}exp(-\pi^2 t)sin\pi x + exp(-4\pi^2 t)sin2\pi x)}{1 + \frac{1}{4}exp(-\pi^2 t)cos\pi x + \frac{1}{2}exp(-4\pi^2 t)cos2\pi x} \tag{11}$$

At time t=0.0, our initial conditions in (1) are thus found, from (11), to be

$$u(x,0) = \frac{2\pi(\frac{1}{4}sin\pi x + sin2\pi x)}{1 + \frac{1}{4}cos\pi x + \frac{1}{2}cos2\pi x}\,. \tag{12}$$

This corresponds to a positively-travelling velocity wave in the lefthand portion of our 1-D space and a negatively-travellirg velocity wave in the righthand region. With these initial velocity waves, the final velocity at t=1.0 is approximately zero.

936-32

Nonlinear Algorithm Steps	
STEP 1	Set initial conditions, $u^{n,r} = u^{0,0}$
STEP 2	So ve LAE below for $u^{n+1} = u^{n+1,r} = u^{n+1,0}$
	$D(u^n)u^{n+1} = e^n$ Eq. (5)
	where $D(u^{n+1})$ in Eq. (4) is approximated by $D(u^n)$ in Eq. (5)
STEP 3	Refine solution (by Newton-Raphson)
	Solve LAE below for $u^{n+1,r+1}$
	$J^{n+1,r}(u^{n+1,r+1} - u^{n+1,r}) = -R^{n+1,r}$ Eq (6)
	where
	Residual vector = $R^{n+1,r} = [D(u^{n+1,r})u^{n+1,r} - e^{n,0}]$
	Jacobian = $J^{n+1,r} = \frac{\partial R^{n+1,r}}{\partial u^{n+1,r}} = A + \frac{1}{2}\Delta t(B(u^{n+1,r}) + B(u^{n+1,r}) + C)$
	$e^{n,0}$ is a function of the converged response from the previous time step $u^{n,0}$
STEP 4	Update J and R and form new Eq. (6), $r \rightarrow r+1$
	Solve new Eq. (6) for $u^{n+1,r+2}$, etc.
	Repeat until convergence, $\mid u^{n+1,r+1} - u^{n+1,r} \mid < \epsilon_{NR}$
	Result is u^{n+1} at present time
STEP 5	Go to Step 2
	Use Step 4 result u^{n+1} to solve for u^{n+2}, etc.
	Repeat Step 2 and iterative Steps 3 and 4 until reach time t=1.0

Table 1: Outline of the alorithm steps used to solve Eq. 4

936-32

3.2 LU and Gauss-Seidel Methods

The major computational step in the algorithm is the solution of the LAEs in (5) and (6). We consider both direct and iterative LAE solution algorithms on the OLAP. The LU direct decomposition method is easily implemented on the OLAP of Fig. 1 and is the direct solution algorithm of choice for solving LAEs, since it requires only one vertical channel, i.e. $M = 1$, for any size matrix [5]. Pivoting is not required, because our matrix elements are within an order of magnitude of each other. We choose the Gauss-Seidel method as the iterative LAE solution to be considered, because it is easily implemented and does not require the calculation of an acceleration parameter. The choice of an acceleration parameter is difficult when solving nonlinear PDEs because the matrix $B(u)$ changes with each time step iteration and therefore the optimum acceleration parameter changes also. An attempt to calculate an acceleration parameter at each iteration step would increase the processing time.

4. Simulation Results

In this Section, we detail the results of our case study simulations. Our purpose is to show that the OLAP of Fig. 1 can be used to solve CFD problems, that it can perform bit and matrix partitioning, operate in a negative base, operate in different radices and that it can solve nonlinear matrix equations and LAEs by direct and iterative methods.

Our initial simulations used standard double precision, N=64 bit, fixed point number representation (i.e., for radix -2 we assume N=64 channels in the AO cell and N=64 detectors at P_3 in Fig. 1). For larger radices, fewer channels and detectors are necessary to handle the equivalent of 64-bit precision. With radix -2 encoding, each word is represented by an $N = 64$ bit word. For radix -4 encoding, 32 digit words were used and 22 digit words were used with radix -8. In each case, half of the digits represented the fraction portion of each word and half represented the integer portion. Note that increasing the magnitude of the radix allows a corresponding decrease in the number of bits necessary to achieve the same precision. With 22 digit words in radix -8 there is slightly greater precision than 64 bits. However, 22 digits gave the smallest difference, while keeping at least 64 bits and the same number of digits in the fraction and integer portions of the words. Note that speed increases as the radix is increased, since N is less and $NT_1 = T_2$. However, the number of output A/D bits increases as the radix increases. In our studies, the number of AO cell channels and detector channels was assumed to equal the number of digits required (not bits). The number of processing channels M (the number of LDs) varies with the algorithm. For the LU algorithm, we require only one (M=1) processing channel. For the Gauss-Seidel algorithm, we require M=3 processing channels (since the matrix is tridiagonal).

Figure 2 shows a plot of the analytical (solid line) and the optically calculated finite element (dashed line) velocities for five of the time steps and for radix -4. Identical plots resulted for radices -2 and -8 (not shown). Both the analytical and optical solution curves are nearly identical at each time step. Table 2 lists the results of the LU and Gauss-Seidel algorithms for three different negative radices. The results include the standard deviation σ (of the difference between the analytical and optical solutions), the number N_{vip} of VIPs required in each algorithm and the processing time in units of T_1. The standard deviation was calculated for each time step and averaged over all 200 time steps, resulting in the σ values listed in Table 2. The standard deviation is a maximum of σ=0.034 at t=0.005 and monotonically decreases to σ = 0.00000153 at time t=1.0, giving an average value of 0.0018. We see that σ is the same for all radices and both LAE solution algorithms. This is expected, since the word precision is nearly identical for all three radices and the stopping criterion in the Newton-Raphson iterations is the same very small value $\epsilon_{NR} = 10^{-5}$. This value was initially selected since with 64 bits, the Newton-

936-32

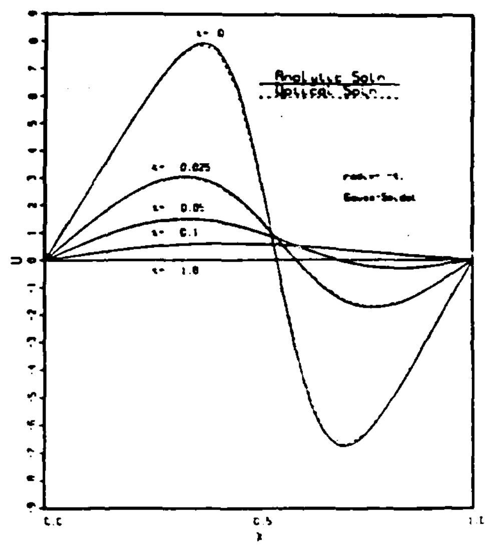


Figure 2: Plot of analytical vs finite element solutions as generated by the optical simulator

Raphson iterations provided this accuracy after only $r=3$ Newton-Raphson iterations. We assigned the same ϵ stopping criterion to the iterative Gauss-Seidel algorithm. Thus, the same σ is expected for all algorithms and radices. A measure of the percent error between the analytical and optically calculated solutions is not feasible due to numerous zero-crossings in the curves which tend to exagerate percent errors. Therefore, we use only the standard deviation between the analytical and optical solutions as our error measure. However, one can obtain a rough estimate of a percent error by dividing the average standard deviation value from Table 2 by the maximum velocity value (of the analytical solution) from Fig. 2. The resulting quotient relates the errors between the analytical and optical solutions to the velocities from which the errors are calculated and ideally, should be less than 1% (due to the Δt chosen in Section 3.1). From Fig. 2, the maximum velocity is seen to be ~8 m/sec, thus our percent error measure is σ/u_{max} = 0.0018/8. x 100 % = 0.02 %, which is sufficiently small. The 1% design value in Section 3.1 is a worst case value and assumes an infinite processor accuracy and is independent of the algorithm used. Thus, our much better accuracy obtained (0.02% versus 1%) is not unexpected. Because percent error is not a rigorous or standard CFD error measure, we will use only the standard deviation as our error measure in future work.

The fifth column of Table 2 lists the number (N_{vip}) of optical vector-inner-products required in each algorithm. For each of the different radices, the iterative Gauss-Seidel method requires approximately 5% more VIPs than the LU direct method. This is not a general result and is due to the fact that the Gauss-Seidel stopping criterion ϵ_{GS} was set equal to the Newton-Raphson stopping criterion (which was very small, $\epsilon_{NR} = \epsilon_{GS} = 10^{-5}$). For this case study, only $r=3$ Newton-Rapshon iterations were required for convergence. Thus, the algorithm rapidly refined the initial LU and Gauss-Seidel solutions in Step 2 to the proper result. The number of Gauss-Seidel iterations was large (but not excessive), i.e., between 19 and 21 iterations in each of the three Newton-Raphson iterations. This was necessary since the Gauss-Seidel solution had a very small stopping criterion. Similarly, $r=3$ Newton-Raphson iterations resulted (rather than $r=1$ for ϵ_{NR}=0.1 and ϵ_{GS}=0.01 as we will show later) because of the small ϵ_{NR} stopping requirement.

The last column lists the processing time T_p for this case study on the system of Fig. 1. The general

936-32

radix	*LAE algorithm*	*no. of digits, N*	*std. dev. σ*	*no. of VIPs (N_{vip})*	*processing time (N_{vip} x N x T_1) , $T_1 = 15$ nsec*
−2	LU decomp.	64	0.0018	110,125	$\sim 7.0 \times 10^6 \times T_1 = 105.7$ *msec*
	Gauss-Seidel	64	0.0018	115,083	$\sim 7.4 \times 10^6 \times T_1 = 110.4$ *msec*
−4	LU decomp.	32	0.0018	110,125	$\sim 3.5 \times 10^6 \times T_1 = 52.9$ *msec*
	Gauss-Seidel	32	0.0018	115,083	$\sim 3.7 \times 10^6 \times T_1 = 55.2$ *msec*
−8	LU decomp.	22	0.0018	110,125	$\sim 2.4 \times 10^6 \times T_1 = 36.3$ *msec*
	Gauss-Seidel	22	0.0018	115,083	$\sim 2.5 \times 10^6 \times T_1 = 38.0$*msec*

Table 2: Initial simulation results: double-precision operation in the region $0.0 < t < 1.0$ with $\epsilon_{NR} = 10^{-5}$ and $\epsilon_{GS} = 10^{-5}$

radix	*LAE algorithm*	*no. of digits, N*	*std. dev. σ*	*no. of VIPs (N_{vip})*	*processing time (N_{vip} x N x T_1) , $T_1 = 15$ nsec*
−2	LU decomp.	64	0.015	9420	$\sim 602.9 \times 10^3 \times T_1 = 9.0$ *msec*
	Gauss-Seidel	64	0.015	5472	$\sim 350.2 \times 10^3 \times T_1 = 5.2$ *msec*
−4	LU decomp.	32	0.015	9420	$\sim 301.4 \times 10^3 \times T_1 = 4.5$ *msec*
	Gauss-Seidel	32	0.015	5472	$\sim 175.1 \times 10^3 \times \mathrm{T}_1 = 2.6$ *msec*
−8	LU decomp.	22	0.015	9420	$\sim 207.2 \times 10^3 \times T_1 = 3.1$ *msec*
	Gauss-Seidel	22	0.015	5472	$\sim 120.4 \times 10^3 \times T_1 = 1.8$ *msec*

Table 3: Double-precision operation in the region $0.0 < t < 0.1$ with ϵ_{NR}=0.1 and $\epsilon_{GS} = 0.01$

expression for T_p is

$$T_p = N_{vip}T_2 = N_{vip}NT_1 \tag{13}$$

where N_{vip} is the total number of VIPs performed by the OLAP in the 200 time steps from $t = 0.0$ to $t = 1.0$. A realistic value of T_1 has been shown [5] to be $T_1 = 15$ *nsec*. This value was used to obtain the numerical results in the last column of Table 2. Note that different values of N are used with different radices. The speed advantage when processing with higher radices is obvious from these numbers. This and the ability of the optical processor to solve the nonlinear dynamic PDE by both algorithms are key points.

The larger N_{vip} and T_p for the Gauss-Seidel versus the LU algorithm are not typical (due to small ϵ_{NR} and ϵ_{GS} used). These small ϵ were chosen since three Newton-Raphson iterations produced this accuracy. As we shall see, this low ϵ value is not necessary. Thusfar, N=64 bit precision has been assumed. To decouple the algorithm from the processor, we now consider the number of bits required and the ϵ values needed. To reduce simulation time, we only consider the velocity in some time interval, rather than from t=0 to steady state (t=1.0). We chose to investigate the velocities over the initial time interval from t=0.0 to 0.1, where there is the most activity. This applies to all further tests. As

936-32

our error measure we use σ. Although not shown, the value of σ in the region $0.0 < t < 0.1$ with $\epsilon_{NR} = \epsilon_{GS} = 10^{-5}$, is σ=0.015. We now consider selecting a larger ϵ to achieve σ=0.015 (which provides sufficient accuracy). We measured σ as ϵ_{NR} was increased (keeping N=64 bits) and found σ=0.015 for $\epsilon_{NR} \leq 0.1$. We thus selected ϵ_{NR}=0.1. This larger ϵ_{NR}=0.1 value was sufficient, since the initial accuracy of the u^{n+1} estimate from (5) was sufficient that the Newton-Raphson algorithm could refine it in $r = 1$ Newton-Raphson iteration to $\epsilon_{NR} = 10^{-3}$. For low ϵ values, setting $\epsilon_{GS}=\epsilon_{NR}$ gave suitable results. For the larger present ϵ_{NR} value, $\epsilon_{GS}=0.1\epsilon_{NR}$ is needed (since the Newton-Raphson algorithm cannot easily refine coarse initial estimates). Thus, by specifying ϵ_{GS}=0.01, we achieve a sufficiently accurate initial estimate that can be refined by Newton-Raphson. Otherwise, Newton-Raphson does not converge. Thus, we select ϵ_{GS}=0.01 and ϵ_{NR}=0.1. These appear to be new results. The number of Gauss-Seidel iterations to solve the different LAEs was significantly reduced with the larger ϵ_{GS}=0.01. Specifically, with ϵ_{GS}=0.01, only eight Gauss-Seidel iterations were necessary to solve for the initial u^{n+1} in (5) and a maximum of ten Gauss-Seidel iterations where necessary to solve the Newton-Raphson LAE in (6). At larger time steps, i.e. as t approaches 0.1, the number of Gauss-Seidel iterations required was reduced. With ϵ_{NR} = 0.1 and ϵ_{GS}=0.01, fewer VIPs are thus necessary for Gauss-Seidel than for LU decomposition, as Table 3 shows. The data in Table 3 for 64 bits, ϵ_{NR}=0.1, ϵ_{GS}=0.01 and $0.0 < t < 0.1$ show the same σ accuracy (our σ=0.015 design goal) for all algorithms and fewer VIPs necessary with Gauss-Seidel than with LU.

In further tests, we found that if ϵ_{GS} was not small enough with respect to ϵ_{NR}, then the error in the Gauss-Seidel solution to (6) could not be improved by the Newton-Raphson iterations and ϵ_{NR} could not be satisfied. Specifically, the Newton-Raphson iterations could not converge. For example, with ϵ_{NR}=0.1 and ϵ_{GS}=0.8 (i.e. with a more accurate Newton-Raphson stopping criterion), no convergence was obtained. A useful (and logical) guideline is to make both ϵ values about the same or to set ϵ_{NR} to be ten times larger than ϵ_{GS} (for larger ϵ_{NR}=0.1 values). Otherwise, Newton-Raphson cannot correct for Gauss-Seidel errors that occur with a larger ϵ_{GS}. Selecting ϵ is a trial and error procedure in all Newton-Raphson applications (whether implemented digitally or optically). This appears to be one of the first quantitative details of the design issue in CFD.

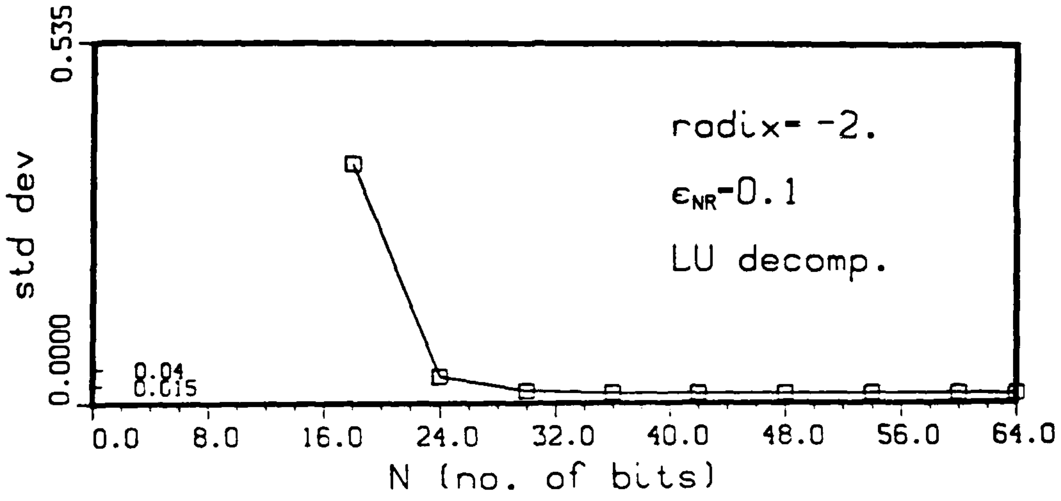
With ϵ_{NR}=0.1 and ϵ_{GS}=0.01 fixed, we now consider the minimum number of bits that each algorithm requires in order to achieve the same σ=0.015 reported in Table 3. Figures 3(a) and 3(b) show σ versus N for the two algorithms with ϵ_{NR}=0.1 and ϵ_{GS}=0.01. As seen, both algorithms perform similarly (as expected) and with approximately $N \geq 30$ bits, we find no more error than with N=64. At N=24 bits, σ increases significantly (with σ=0.04 for N=24). Thus, we now fix N=24 bits, ϵ_{NR}=0.1 and ϵ_{GS}=0.01.

We ran the case study with 24 bits, ϵ_{NR}=0.1 and ϵ_{GS}=0.01. The results are shown in Table 4. The value of σ (0.04) is larger than in Table 3 (σ = 0.015) because we chose to operate at the N=24 bend in the curves in Figs. 3(a) and 3(b). The number of VIPs is the same for all radices (for a given algorithm) because the precision is the same (24 bits) for all the radices. By increasing the number of bits to 30, σ is reduced to 0.015, as shown in Table 5. The number of VIPs remains the same as in Table 4, as expected. Therefore, the minimum number of bits that maintains σ=0.015 is approximately 30 bits.

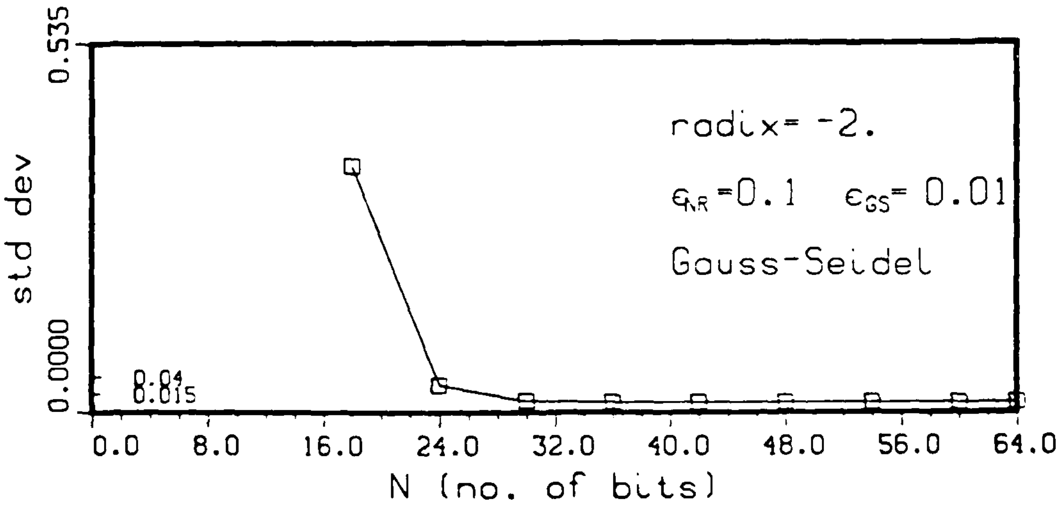
5. Conclusions

We have shown that the OLAP can solve nonlinear dynamic PDEs using iterative and direct algorithms, with a negative base and with non-binary radices. Our primary purpose was to demonstrate the new use of an OLAP in the solution of CFD problems. However, other data can be advanced. We have quantified that the processor operates faster when higher radices are used. We have also advanced a new

936-32



(a)



(b)

Figure 3: Plot of σ vs N for (a) LU decomposition with ϵ_{NR}=0.1 and (b) ϵ_{NR}=0.1 and ϵ_{GS}=0.01 for Gauss-Seidel.

936-32

radix	*LAE algorithm*	*min. no. of digits, N_{min}*	*std. dev. σ*	*no. of VIPs N_{vip}*	*processing time (N_{vip} x N_{min} x T_1) with $T_1 = 15$ nsec*
−2	LU	24	0.04	9420	3.4 *msec*
	Gauss-Seidel	24	0.04	5472	1.9 *msec*
−4	LU	12	0.04	9420	1.7 *msec*
	Gauss-Seidel	12	0.04	5472	1.0 *msec*
−8	LU	8	0.04	9420	1.1 *msec*
	Gauss-Seidel	8	0.04	5472	0.7 *msec*

Table 4: Minimum precision (N=24) operation in the region $0.0 < t < 0.1$ with $\epsilon_{NR} = 0.1$ and $\epsilon_{GS} = 0.01$

radix	*LAE algorithm*	*min. no. of digits, N_{min}*	*std. dev. σ*	*no. of VIPs N_{vip}*	*processing time (N_{vip} x N_{min} x T_1) with $T_1 = 15$ nsec*
−2	LU	30	0.015	9420	4.2 *msec*
	Gauss-Seidel	30	0.015	5472	2.5 *msec*
−4	LU	15	0.015	9420	2.1 *msec*
	Gauss-Seidel	15	0.015	5472	1.2 *msec*
−8	LU	10	0.015	9420	1.4 *msec*
	Gauss-Seidel	10	0.015	5472	0.8 *msec*

Table 5: Minimum precision (N=30) operation in the region $0.0 < t < 0.1$ with $\epsilon_{NR} = 0.1$ and $\epsilon_{GS} = 0.01$

936-32

procedure to determine the stopping criteria for nonlinear algorithm solutions such as Newton-Raphson and a new procedure to determine the number of bits of precision required in the processor. The OLAP can solve CFD problems in reasonable times (as we have quantified). We also showed that similar ϵ values should be used for all iteration steps. We now advance other discussion remarks.

For this case study, the iterative Gauss-Seidel algorithm is faster than the direct LU decomposition algorithm. This occurs when ϵ is properly chosen. As the dimensionality p of the matrix increases, the number of VIPs required in LU will rapidly increase ($N_{vip} \propto p^3$), whereas the number of Gauss-Seidel iterations will not increase as rapidly ($N_{vip} \propto kp$), where k relates to the number of iterations required. Typical CFD problems have $p > 700$. Thus, the Gauss-Seidel algorithm will be over 50,000 times faster (assuming an average k=10). We note that iterative techniques are popular CFD problem solution algorithms for this reason. The present case study is a smooth flow problem. For all such problems, we expect the same trend for Gauss-Seidel to be superior. As p increases, Gauss-Seidel gains even more preference. For our case study, the dynamic range of the matrix data was $\sim$ 40 to 1, but 30 bits are still necessary to adequately represent the data and to perform the necessary operations. With a larger dynamic range, pivoting may be required in LU, further complicating its implementation. With other CFD problems requiring more Gauss-Seidel iterations, it is realistic to expect that we can run the processor at low accuracy for a number of iterations and then switch to higher accuracy for the last few iterations.

We should distinquish between the number of bits necessary and optical systems errors. Specifically, both the Gauss-Seidel and LU algorithms require the same number of bits, where we assume that the number of bits given are all accurate. If we consider optical system errors, then there is an additional preference for the Gauss-Seidel algorithm. This arises since the LU algorithm is more sensitive to processor errors (which can occur in any bit, e.g. the most-significant bit as well as the least-significant bit). Iterative algorithms such as Gauss-Seidel are known to be forgiving of such errors. We have shown that Newton-Raphson cannot correct LAE errors that are too large. Thus, large LU errors (that could occur within an optical processor) will be disastrous. However, the Gauss-Seidel algorithm can correct such optical system errors (with an increase in the number of iterations). The combination of iterative Gauss-Seidel and Newton-Raphson algorithms with similar ϵ is thus very attractive. Since the error in LU is not known (it can be in the most significant or least significant bits), we cannot adjust ϵ in Newton-Raphson to correct it. With a Gauss-Seidel / Newton-Raphson algorihm set, there is much better reason to expect Gauss-Seidel to converge (by including a convergence parameter, the number of iterations and convergence will be even better).

References

[1] "Special issue on optical computing,". *Proc. IEEE*, Vol. 72, No. 7, July 1984.

[2] E. Swartzlander, "The quasi-serial mutiplier," *IEEE Trans. Comput.*, vol. C-22, pp. 317–321, April 1973.

936-32

[3] H. Whitehouse and J. Speiser, “Linear signal processing architectures,” in G. Tacconi, ed., (*Aspects of Signal Processing - Part II*) (Boston, MA) pp. 669–702 Boston, MA 1976. NATO Advanced Study Institute.

[4] D. Psaltis, D Casasent, D. Neft and M. Carlotto, “Accurate numerical computation by optical convolution,” *Proc. SPIE*, vol. 232, pp. 151–156, April 1980.

[5] D. Casasent and B. Taylor, “Banded-matrix high-performance algorithm and architecture,” *Applied Optics*, vol. 24, pp. 1476–1480, May 1985.

[6] C. Perlee and D. Casasent, “Negative base encoding in optical linear algebra processors,” *Applied Optics*, vol. 25, pp. 168–169, January 15 1986.

[7] T. J. Chung, *Finite Element Analysis in Fluid Dynamics.* McGraw-Hill Publishing, 1978.

[8] D. Casasent and S. Riedl, “Time and space integrating optical laboratory matrix-vector array processor,” *Proceedings of the SPIE*, vol. 698, pp. 151-156, August 1986.

[9] B. Taylor and D. Casasent, “Error-source effects in a high-accuracy optical finite-element processor,” *Applied Optics*, vol. 25, pp. 966–975, 15 March 1986.

[10] B. Taylor and D. Casasent, “Optical matrix-vector processing for finite element problems,” *Proceedings of the SPIE*, vol. 939, (Orlando, FL), April 1988.

CHAPTER 6:

APPLICATION OF OPTICAL PROCESSING TO ADAPTIVE PHASED ARRAY RADAR

936-16

Presented at the SPIE Conference in Orlando, Apr. 1988

Application of Optical Processing to Adaptive Phased Array Radar

C.W.Carroll
Mellon Institute, Computer Engineering Center
and
B.V.K.Vijaya Kumar
Electrical and Computer Engineering Department

Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

The results of our investigation of the applicability of optical processing to Adaptive Phased Array Radar (APAR) data processing will be summarized. Subjects that are covered include: (i) new iterative Fourier Transform based technique to determine the array antenna weight vector such that the resulting antenna pattern has nulls at desired locations, (ii) obtaining the solution of the optimal Wiener weight vector by both iterative and direct methods on two laboratory Optical Linear Algebra Processing (OLAP) systems, and (iii) an investigation of the effects of errors present in OLAP systems on the solution vectors.

1. INTRODUCTION

In modern radar signal processing an array of antennas is used to collect radar signals instead of a single antenna[1]. These individual signals are then combined to produce the antenna array output. To allow control over the shape of the farfield antenna response, the amplitude and phase of the individual signals are normally modified prior to forming the array output. This operation is equivalent to multiplying the complex envelope of each signal by a complex weight factor. Such a phased array antenna has pattern nulls which can be steered to attenuate strong jammer signals that would leak through the sidelobes of a fixed pattern antenna.

In Adaptive Phased Array Radar (APAR) signal processing, we seek algorithms to determine the weights which minimize the jammers' effects while maintaining the required directivity. Unfortunately, along with its added capabilities, the APAR problem comes with a much higher computational burden when compared with single antenna systems. Because of the inherent speed and parallelism of optical processors, their application to this computationally intensive problem should be investigated. Our goal in this research effort is to evaluate some of the existing APAR algorithms for their suitability to optical implementation and propose novel algorithms that may not have electronic counterparts. We must look for algorithms that require repetitive tasks which result in acceptable optical data flow and that are tolerant of the errors present in an analog optical system. Any purely optical APAR algorithm we propose must utilize the capabilities of optical processors (such as the instantaneous Fourier transform).

1.1. Adaptive Array Processing

The received signal vector x is comprised of random antenna noise and the sum of the received signals from the I sources for each of the K antenna elements in the array. The array output $y = w^H x$ is the weighted sum of the received signals where w is the weight vector and ^{H} denotes the Hermitian or conjugate transpose. Figure 1 indicates how APAR processing can be divided into communications and tracking/estimation problems.

When using an antenna array in an application such as communications, the primary goal of many adaptive processing schemes is to enhance the reception of the desired signal while suppressing unwanted signals. The optimal weight vector w_{opt} for this problem can be shown[2] to be the solution to a set of Linear Algebraic Equations (LAEs) of the form

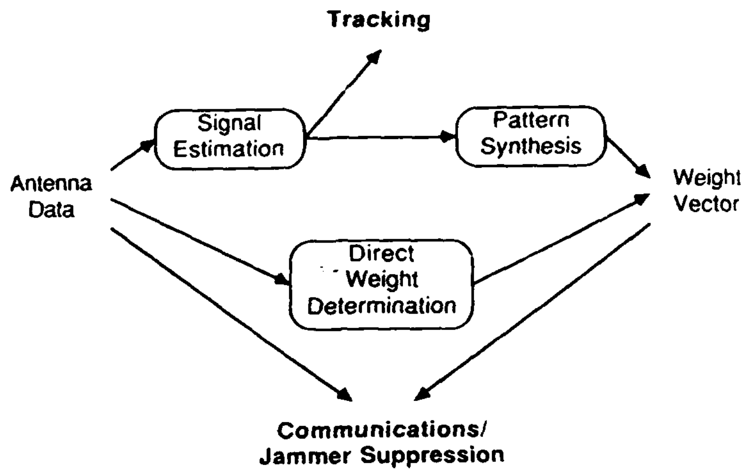


Figure 1: Overview of APAR Processing Tasks

$$\mu R_{xx} w_{opt} = r_{xd}. \tag{1}$$

where μ is a complex scalar, R_{xx} is the received signal covariance matrix and r_{xd} is the correlation between the received signal vector and the desired signal. It should be noted that R_{xx} is Hermitian and thus has real, non-negative eigenvalues.

An intermediate goal of antenna array processing is the estimation of the farfield signal environment. This information can include the number of signal sources, their individual signal powers, their locations, and any correlation between sources. Knowledge of this information is useful for detecting and tracking objects in applications such as air traffic control, satellite tracking, and weapons control. Knowledge of the signal environment can then be used to determine which directions to null. Pattern synthesis techniques can then be used to generate the appropriate weight vector.

Given the information obtained from a source estimation processor, one is faced with the problem of computing an appropriate weight vector. Normally, these requirements can be condensed into a set of constraints such as the desired look direction gain and the directions towards which nulls are to be steered. A major computational task is the calculation of a weight vector which satisfies these constraints. These constraints can be used along with a minimization of an array performance measure such as minimizing the array output power.

1.2. Optical Processors

Relevant prior work using simple optical architectures for adaptive processing includes an adaptive signal predictor[3], a sidelobe canceller[4] which uses correlations among auxiliary antennas to modify the sidelobes of a main antenna, a linear predictor[5], and an adaptive filter[6]. These have made use of optical systems to perform processing via continuous-time signal correlations, whereas the techniques we are investigating involve more numerical processing of discrete-time data.

Since it is possible to formulate the APAR problem in terms of vectors and matrices, it is useful to study how it can be solved using the Optical Linear Algebra Processors (OLAPs) available for our use. An exhaustive discussion of OLAPs is available elsewhere[7, 8]. We confine our attention to methods appropriate for general matrices rather than those with special structures such as Toeplitz or banded matrices. Other investigators have also studied optical numerical data processing with reference to the APAR problem[9, 10]. The use of alternate number representation schemes, such as the residue number system[11, 12], have been published. Additional methods of increasing processing accuracy through the use of multimode architectures have also been presented[13]

The availability of existing laboratory OLAP systems offers a unique opportunity to test prospective APAR algorithms. While computer simulations of optical processors are useful for running preliminary tests, the availability of experimental systems allows a more complete characterization of results to be made. There are two optical systems which have been used through this research effort to obtain the solution of equation 1. Since the initial development of these systems was not part of this research effort, we only reference the experimental systems here, and additional information may be found elsewhere[14, 15].

1.3. Overview of Paper

The remainder of this paper describes specific areas addressed by this research. Section 2 covers an algorithm based upon the repetitive use of Fourier Transforms for the pattern synthesis problem. We next address the weight determination problem in Section 3 using iterative methods to solve equation 1 on an analog optical M/V processor. A method useful in the characterization of systems capable of changing from high speed, low accuracy to higher accuracy, slower speed while implementing an iterative algorithm is presented in Section 4. The solution of the equation for the optimal weight vector using an encoded OLAP with a matrix decomposition algorithm is examined in Section 5. We summarize the research in Section 6.

2. ITERATIVE FOURIER TRANSFORM PATTERN SYNTHESIS

This section proposes and examines an algorithm which can provide a weight vector that places nulls at desired angles and is well-suited to optical processing. For narrowband signals being received on an arbitrary, planar, continuous array, the angular gain vector g and the antenna weight vector w are related through a Fourier Transform. To simplify the discussion, let us limit ourselves to the case of the linear array with equally spaced antenna elements. This situation can be conveniently written in terms of a Discrete Fourier Transform. To place antenna response nulls in the jammer directions, we specify $(g)_q = 0$ at the q values corresponding to the jammer angles. The Inverse Fourier transform of g then yields the weights w. However, due to the limit in the total number of antenna elements, K, the angular resolution in q is poor. To improve this sampling, we can pad w with zeros so that its new length is N and then take the transform. In the final weight vector applied to the antenna output, the contribution of these extra, virtual elements must be zero since they do not correspond to available antenna elements. This provides two sets of constraints: (i) gain g equal to zero for each jammer location and (ii) antenna weights for the virtual antenna elements must equal zero. In addition to these, a normalization constraint, such as forcing the array gain energy or array weight vector energy to a constant, may be required to prevent the output from decaying to zero.

A method of finding a solution to match constraints in both the time and frequency domains is a generalization of the Gerchberg-Saxton algorithm[16]. It is known as the Error Reduction algorithm and Fienup[17] has provided a summary of its use in the retrieval of phase information from intensity image data. The algorithm involves repeated FT's where the resulting functions are forced to meet the constraints first in one domain and then in the other. A block diagram of this algorithm is presented in Figure 2. Starting with the weight vector w, it is first transformed into the angular gain domain by the DFT block. The angular constraints are then applied to the gain vector followed by the inverse DFT to return to the weight domain. Finally, the antenna constraints are imposed to complete one iteration of the algorithm.

The repetitive use of the Fourier transform operation in this algorithm is well-suited to implementation on an optical system. A simple example of such an optical implementation has previously been proposed for use in the extrapolation of bandlimited images[18]. The basic system uses an optical FT lens for performing the Fourier transforms, masks to impose the constraints in each domain, and mirrors to provide the iterative feedback. Because of its two dimensional nature, this implementation can be easily extended to pattern synthesis for 2-D arrays of an arbitrary geometry.

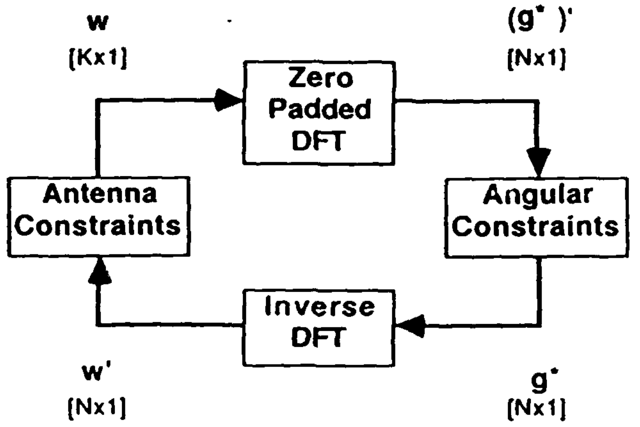


Figure 2: Block Diagram of the Error Reduction Algorithm[17]

We have previously presented our investigation into the behavior of this algorithm as applied to the discrete case[19]. The algorithm is first written in matrix/vector form and then its transient and steady-state responses may be derived. Under specific investigation are the values of the weights and the angular domain constraint error as functions of the number of iterations of this algorithm. To assist in this analysis, a new matrix operator has been introduced which greatly simplifies the required derivations. We have derived the conditions required for the algorithm to converge and have shown that when these conditions are met, the steady-state error theoretically approaches zero. We are eventually interested in the location and depth of the nulls, along with the rate of convergence of this algorithm. Computer simulation and numerical evaluations of the analytical results for several APAR test scenarios have been run and verify our analysis.

3. ITERATIVE LAE SOLUTION USING OLAPS - THE STEEPEST DESCENT METHOD

This section examines the use of some of the classical, gradient based algorithms for the solution of a set of LAEs[20]. As discussed in Section 1.1, the fact that these algorithms require a large number of matrix/vector or vector/vector multiplications immediately suggests their implementation on Optical Linear Algebra Processors. Systems that have been optimized for a simple matrix operation, such as M/V multiplication, are thus well suited for this application. The iterative algorithms appear promising due to their simple, repetitive nature. However, they also suffer from the drawback that the number of iterations required to reach a solution with a given accuracy is not known a priori. In this section we describe the digital simulations and laboratory experimentation used to evaluate the applicability of these methods to the optical processing systems we have available.

3.1. Steepest Descent Algorithm

A goal of optimal adaptive array processing is to obtain the weight vector in equation 1 which minimizes the expected squared error between the array output and the desired signal. For the linear array this error is a quadratic function of the weight vector w. Descent methods based upon the local gradients of the error surface can be used to find the w_{opt} that satisfies this criterion. The method of steepest descent is characterized by the weight update equation

$$w(n+1) = (I - 2\beta R_{xx})w(n) + 2\beta r_{xd}\,, \qquad (2)$$

or

$$w(n+1) = w(n) - 2\beta R_{xx}w(n) + 2\beta r_{xd}\,. \qquad (3)$$

where β is the step size parameter which controls the rate of convergence. The limits on β to ensure convergence[2] are $0 < \beta < 1/($maximum eigenvalue of $R_{xx})$. The algorithms in equations 2 and 3 use the covariance matrix R_{xx} and determine the updated weight vector $w(n+1)$ using matrix/vector multiplications with the weight vector $w(n)$. Thus, we can use the advantages of the high-speed optical M/V processors for these algorithms.

Computer simulations have been run to test these algorithms in the presence of optical errors using a precomputed R_{xx}. The starting weight vector was chosen to provide unity gain at zero degrees, where the desired signal is located. In both cases, β was chosen to be well within the range of values allowable for convergence. This set of simulations was run using a simple error model assuming five percent optical (both static and dynamic) errors. When run with no processor errors, both forms of the algorithm reached steady-state levels after less than fifty iterations. However, in the presence of errors, the two updating equations behave differently. Equation 2 begins to adapt correctly, however the jammer plus noise power starts to increase for subsequent iterations. At the same time, the signal power decreases, forming a notch that nulls the desired signal and then increases, following the trend of the jammer plus noise power. This behavior indicates a steadily increasing antenna gain and is unacceptable. Simulations using adaptive equation 3 maintain the signal power nearly constant at the optimal level. The jammer plus noise power stabilizes at a level slightly above the optimal level. Through the use of different levels of processor error, it was observed that the major effect of the static error vector is the rate at which the power obtained with equation 2 increases and in the offset from the optimal level for equation 3. The temporal error vector seems to cause the rapid fluctuations about these basic trends.

The algorithms in equations 3 and 2 are analytically identical, however, our simulations have shown that the algorithm in equation 3 is much less sensitive to noise in the optical M/V system than the algorithm in equation 2. Even though equation 3 requires an extra vector addition in each iteration, its superiority over the algorithm in equation 2 for optical implementation has been demonstrated. While both algorithms work well in the absence of processing errors, we have shown the importance of reevaluating the existing algorithms for optical implementation.

3.2. Experimental System

Although computer simulations can provide useful information concerning the performance of a system, they can only be as accurate as the underlying system models on which they are based. The availability of a laboratory optical system provides a unique opportunity to test the performance of an algorithm on an actual optical system. The experimental system used for this portion of the research is a frequency multiplexed acousto-optic (AO) matrix/vector processor. A simplified schematic of this processor[15] is shown in Figure 3. This system had been constructed by other researchers in our laboratories as part of other research projects. Full details on the system and its initial testing are available elsewhere[21].

Initial experiments with the laboratory system indicate that the limited accuracy of the analog system presents the greatest impediment to its use for APAR processing. However, methods to increase the system accuracy through encoding or coherent detection are means to get around this problem.

3.3. Relationship Between Power and Steepest Descent Methods

An interesting relationship that we have not found reported in the APAR literature can be obtained by examining the steepest descent method of equation 2 when $r_{xd}=0$, i.e.,

$$w(n+1) = (I - 2\beta R_{xx})w(n). \tag{4}$$

This case occurs in the estimation problem which has no desired signal. If the weight vector is further constrained to a specific length; i.e. $w^H(n)w(n) = \textit{constant}$ by scaling after each iteration, then this method becomes equivalent to the shifted version of the Power Method[20]. This method can be used to find the

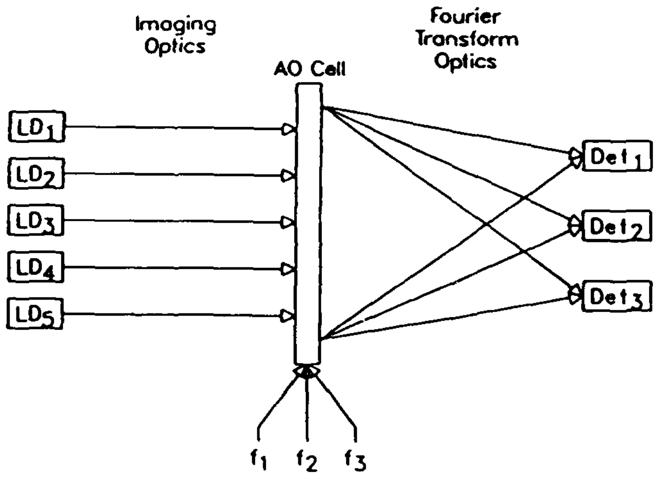


Figure 3: Frequency-Multiplexed Optical M/V Processor[15]

minimal eigenvector of the matrix R_{xx} when $0 < \beta < 1/(\lambda_{min} + \lambda_{max})$, where the minimum and maximum eigenvalues of R_{xx} are denoted by λ_{min} and λ_{max} respectively. Combinations of the minimal eigenvectors of the covariance matrix can be used to obtain estimates of the signal locations and strengths[22]. The importance of this relationship is the manner in which it ties together the spectral estimation methods with the simple, iterative weight determination methods.

4. VARIABLE ACCURACY OLAPS - SPEED/ACCURACY TRADEOFFS

Conventional optical processing has generally been analog in nature and thus the accuracy of the output has been limited. For example, the analog processor discussed in Section 3 may not be accurate enough to obtain a weight vector which provides the required null depths or SNR. More recently, techniques for obtaining higher accuracy through the encoding of numerical data have been applied to optical processors[23, 24]. The increase in accuracy is generally accompanied by a reduction in processing speed or an increase in hardware complexity. With some architectures, it is possible to select processing modes having different accuracies but which run on the same hardware. These systems can be divided into those that can operate in either analog or encoded modes, or those whose level of encoding is variable. The processors shown in Figures 3 and 4 are examples of systems which can operate in several accuracy modes.

Processors supporting several accuracy modes, combined with an iterative algorithm, offer the ability to begin calculations with comparatively fast, low accuracy steps and then finish with slower, high accuracy steps to ultimately provide an accurate solution to a system of linear algebraic equations (LAEs). It is thus useful to be able to determine analytically the effects of processor errors on the solution vector. This allows a direct comparison between processing systems having different accuracies, or between different operating modes of processors which are able to dynamically change their accuracy.

We combine the analytical results for the convergence of iterative methods with the accuracies of the optical processors to analyze the effects of switching from low to high accuracy modes[25]. An important question is whether the ability to switch between low and high accuracy processing modes can provide a useful gain in speed over using just the high accuracy mode, while maintaining the solution accuracy of the high accuracy processing. Through use of a classical analysis[26], we obtain a relationship between the accuracy in the solution obtained using the iterative algorithm and the number of processing iterations. This analysis is applied to simple test cases, representative of APAR problems, and is numerically evaluated to provide a family of curves illustrating the dependence of solution accuracy based upon the basic system accuracy and the number of iterations. These results are then interpreted for specific optical architectures to determine the best case gain in processing speed realized through the use of a variable accuracy processing.

This analysis is useful for the characterization of the convergence properties of the steepest descent method in the presence of processing errors. We have applied this analysis to assist in the determination of the maximum allowable processor error level that can be tolerated while maintaining a desired accuracy in the solution vector. An important consideration is that an algorithm which is converging to the correct solution may eventually reach a point where continued updates degrade the solution. Sometime before this point is reached, the algorithm should be either terminated, or switched into a higher accuracy processing mode. Although a practical method for the online estimation of the switching point is not proposed here, the techniques presented in this section can be used as tools which can be applied to a given processing architecture to estimate the utility of switching accuracy modes. Improvements in processing speed as a result of switching from low to high accuracy, as opposed to merely running the processor at full accuracy, are extremely dependent on the configuration of the processor, the structure of the LAE, and the starting vector. It is possible that a gradual improvement in the processor accuracy may provide an even greater advantage in processing speed. Another possible situation would be to start by using analog processing and then switch to encoded processing to obtain the higher level accuracy. These situations can be evaluated using this technique for specific processors. Decisions made using these relations must be applied with caution since sufficiently precise estimates of the processor error and processing time must be used to obtain practical results.

5. DIRECT LAE SOLUTION USING OLAPS - GAUSS ELIMINATION

One of the most familiar methods of solving a set of LAEs consists of converting the matrix to upper triangular form and then completing the solution through back-substitution[20]. One important advantage of this approach is that the number of steps required to obtain the solution is known a priori and is independent of the actual data. A problem with the matrix decomposition algorithms is that errors introduced early in the computations tend to be magnified in the subsequent steps. This occurs since there is no feedback to allow self-correction of the errors and necessitates the implementation of this algorithm on a high accuracy processor. A version of this algorithm, where the data flow has been designed specifically for a high accuracy optical M/V multiplication system, has previously been reported[27]. We have demonstrated the use of this algorithm on a laboratory optical system to obtain the solution for the optimal weight vector[28]. The specific system is a 10 channel AO linear algebra processor[14] that uses binary encoded numbers to provide greater accuracy. A schematic of this optical system is shown in Figure 4. We have also investigated, through digital simulation, the effects of data truncation to different

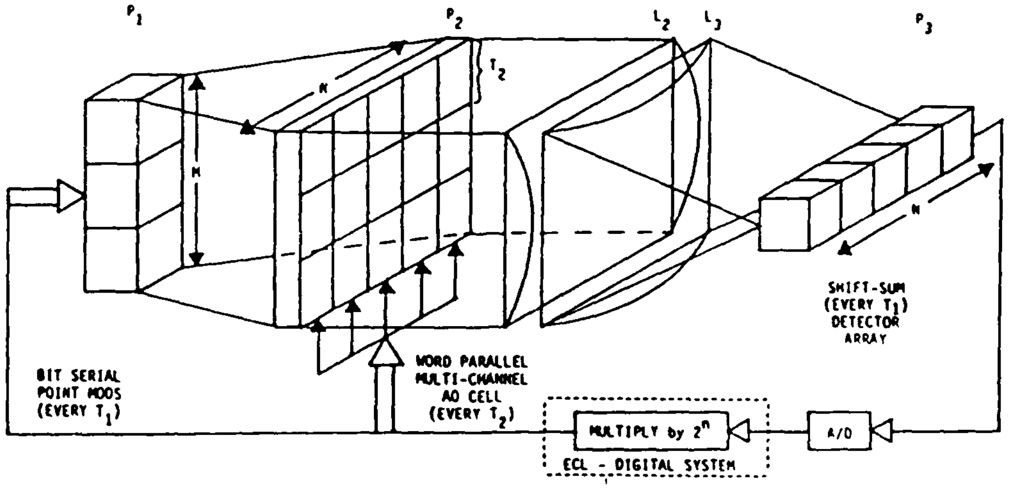

P_1
P_2
L_2
L_3
P_3
T_2
N
M
N
SHIFT-SUM (EVERY T_1) DETECTOR ARRAY
WORD PARALLEL MULTI-CHANNEL AO CELL (EVERY T_2)
BIT SERIAL POINT MODS (EVERY T_1)
MULTIPLY by 2^n
A/D
ECL - DIGITAL SYSTEM

Figure 4: Multi-Channel, Encoded Optical M/V Processor[14]

finite word sizes and the use of various scale factors on the input data. Null depths in excess of 90 dB were obtained for test scenarios using 21 bit processing on the optical system, yielding essentially optimal antenna performance.

6. CONCLUDING REMARKS

In the preceding sections, we have provided an overview of our investigation of several areas where optical data processing techniques are applicable to APAR problems. Although many different topics have been investigated, each addressing a different aspect of optical processing techniques, the unifying theme is optical processing of APAR data.

A new iterative technique has been applied to the pattern synthesis problem. The algorithm provides a weight vector which matches a set of constraints on the placement of nulls in the farfield pattern. The algorithm makes excellent use of the optical Fourier transform property. Numerical evaluations for specific test cases demonstrate successful operation of this algorithm.

The equation for the optimal Wiener weight vector has been solved by both iterative and direct techniques on two laboratory Optical Linear Algebra Processing (OLAP) systems. Iterative techniques were implemented on a low accuracy analog system, while direct techniques were used with high accuracy, encoded systems. Solutions obtained on the encoded optical system have been essentially identical to those obtained using commercially available software routines and floating point arithmetic on a minicomputer. These results clearly demonstrated that optical systems can be applied even when the demands on solution accuracy are high.

Investigations have also been carried out concerning the effects of errors present in OLAP systems on the solution vector obtained through an iterative algorithm. Examples demonstrate the existence of a transition point where further iterations either do not improve the solution or actually degrade the solution. We used these transition points to explore the prospect of running an optical system initially in a high speed, low accuracy mode followed by a lower speed, higher accuracy mode to obtain a high accuracy solution. For the example processor and scenarios that were analyzed, a net increase in processing speed was realized using this technique.

It has been demonstrated through laboratory experiments, theoretical analysis, and computer simulation, that there are several niches where optical signal processing provides useful results for APAR problems.

7. ACKNOWLEDGMENTS

The authors wish to acknowledge the support of this research by the *Air Force Office of Scientific Research*, contract number AFOSR-84-0239.

8. REFERENCES

1. S.P.Applebaum, "Adaptive Arrays", *IEEE Trans. on Antennas and Propogation*, Vol. AP-24, No. 5, September 1976, pp. 585-598.

2. R.A.Monzingo and T.W.Miller, *Introduction to Adaptive Arrays*, Wiley, New York, 1980.

3. A.VanderLugt, "Adaptive Optical Processor", *Applied Optics*, Vol. 21, No. 22, November 15 1982, pp. 4005-4011.

4. W.A.Penn, "Acousto-Optic Adaptive Signal Canceller", *Proc. ICALEO*, Laser Institute of America, 1982, pp. 9-17.

5. J.F.Rhodes, "Adaptive Filter with a Time-Domain Implementation Using Correlation Cancellation

Loops", *Applied Optics*, Vol. 22, No. 2, 15 January 1983, pp. 282-287.

6. D.Psaltis and J.Hong, "Adaptive Acoustooptic Filter", *Applied Optics*, Vol. 23, No. 19, 1 October 1984, pp. 3475-3481.

7. D.P.Casasent, "Acoustooptic Linear Algebra Processors: Architectures, Algorithms, and Applications", *Proc. of IEEE*, Vol. 72, No. 7, July 1984, pp. 831-849.

8. W.T.Rhodes and P.S.Guilfoyle, "Acoustooptic Algebraic Processing Architectures", *Proc. of IEEE*, Vol. 72, No. 7, July 1984, pp. 820-830.

9. E.Pochapsky and D.P.Casasent, "Adaptive Optical Processing with a Linear Acousto-Optic Heterodyning System", *Vol. 886 Optoelectronic Signal Processing for Phased-Array Antennas*, SPIE, January 1988.

10. E.Baranoski and D.P.Casasent, "Optical Processing of Covariance Matrices for Adaptive Processors", *Vol. 886 Optoelectronic Signal Processing for Phased-Array Antennas*, SPIE, January 1988.

11. A.P.Goutzoulis and D.K.Davies, "On the Characteristics of Practical Optical Residue Look-Up Table Processors", *Vol. 827 Real Time Signal Processing X*, SPIE, August 1987, pp. 225-234.

12. A.P.Goutzoulis, "Complexity of Residue Position-Coded Lookup Table Array Processors", *Applied Optics*, Vol. 26, No. 22, 15 November 1987, pp. 4823-4831.

13. H.J.Caulfield, J.H.Gruninger, H.E.Ludman, K.Steiglitz, H.Rabitz, J.Gelfand, and E.Tsoni, "Bimodal Optical Computers", *Applied Optics*, Vol. 25, No. 18, 15 September 1986, pp. 3128-3131.

14. D.P.Casasent and B.K.Taylor, "Banded-Matrix High-Performance Algorithm and Architecture", *Applied Optics*, Vol. 24, No. 10, 15 May 1985, pp. 1476-1480.

15. D.P.Casasent, J.Jackson, and C.P.Neuman, "Frequency-Multiplexed and Pipelined Iterative Optical Systolic Array Processors", *Applied Optics*, Vol. 22, No. 1, 1 January 1983, pp. 115-124.

16. R.W.Gerchberg and W.O.Saxton, "A Practical Algorithm for the Determination of Phase From Image and Diffraction Plane Pictures", *Optik*, Vol. 35, No. 2, April 1972, pp. 237-246.

17. J.R.Fienup, "Phase Retrieval Algorithms: a Comparison", *Applied Optics*, Vol. 21, No. 15, 1 August 1982, pp. 2758-2769.

18. R.J.Marks II, "Coherent Optical Extrapolation of 2-D Band-Limited Signals: Processor Theory", *Applied Optics*, Vol. 19, No. 10, 15 May 1980, pp. 1670-1672.

19. C.W.Carroll and B.V.K.Vijaya Kumar, "Iterative Fourier Transform Phased Array Radar Pattern Synthesis", *Vol. 827 Real Time Signal Processing X*, SPIE, August 1987, pp. 179-186.

20. G.H.Golub and C.F.VanLoan, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1983.

21. J.Jackson, *Development and Performance Evaluation of an Acousto-Optic Linear Algebra Processor,* PhD dissertation, Carnegie-Mellon University, February 1986.

22. M.Wax, T.J.Shan, and T.Kailath, "Spatio-Temporal Spectral Analysis by Eigenstructure Methods", *IEEE Trans. on Acoustics, Speech and Signal Processing,* Vol. ASSP-32, No. 4, August 1984, pp. 817-827.

23. H.J.Caulfield, D.S.Dvore, and J.H.Gruninger, "Efficient Real Number Representation with Arbitrary Radix", *Applied Optics,* Vol. 23, No. 18, 15 September 1984, pp. 3149-3151.

24. B.K.Taylor and D.P.Casasent, "Twos-Complement Data Processing for Improved Encoded Matrix-Vector Processors", *Applied Optics,* Vol. 25, No. 6, 15 March 1986, pp. 956-961.

25. B.V.K.Vijaya Kumar and C.W.Carroll, "Variable Accuracy Optical Matrix/Vector Processors - Speed Accuracy Tradeoffs", *Vol. 752 Digital Optical Computing,* SPIE, January 1987, pp. 179-186.

26. J.W.Goodman and M.S.Song, "Performance Limitations of an Analog Method for Solving Simultaneous Linear Equations", *Applied Optics,* Vol. 21, No. 3, 1 February 1982, pp. 502-506.

27. D.P.Casasent and A.Ghosh, "Direct and Implicit Optical Matrix-Vector Algorithms", *Applied Optics,* Vol. 22, No. 22, 15 November 1983, pp. 3572-3578.

28. C.W.Carroll and B.V.K.Vijaya Kumar, "Adaptive Phased Array Radar Processing on a Multi-Channel, Acousto-Optic, Linear Algebra System: Experimental Results", *Vol. 886 Optoelectronic Signal Processing for Phased-Array Antennas,* SPIE, January 1988.

CHAPTER 7:

DIRECT FINITE ELEMENT SOLUTION ON AN OPTICAL LABORATORY MATRIX-VECTOR PROCESSOR

DIRECT FINITE ELEMENT SOLUTION ON AN OPTICAL LABORATORY MATRIX-VECTOR PROCESSOR

David CASASENT and Steven RIEDL
Carnegie Mellon University, Center for Excellence in Optical Data Processing, Department of Electrical and Computer Engineering, Pittsburgh, PA 15213, USA

Received 28 September 1987

The first optical laboratory system results employing a direct LU decomposition solution of a system of linear algebraic equations are presented for a finite element problem solution. This also represents the first laboratory demonstration of the use of sign-magnitude negative number representation as well as new bit partitioning techniques to increase the accuracy of an optical encoded processor beyond the number of bit channels available.

1. Introduction

Optical matrix–vector processors [1] represent the major elements in artificial neural networks [2], associative processors [3], optical crossbar switches [4], adaptive processors, and general linear algebraic optical array processors [5]. Such systems have seen little laboratory data results. In this paper, we consider the initial use of an optical laboratory matrix–vector processor [6] operating on encoded data to provide the high-accuracy processing required in the solution of a finite element problem. These results offer a new application (finite element solutions) for optical matrix–vector processors and provide the first laboratory data on the direct solution of a system of linear algebraic equations on an optical laboratory processor. Section 2 briefly reviews the optical laboratory processor used [6,7] and our new one-channel LU decomposition algorithm [7]. Section 3 advances the problem case study considered and the algorithm used. Section 4 presents laboratory data obtained and section 5 advances our summary remarks.

2. Laboratory system

Fig. 1 shows the optical processor we consider [7]. It employs M point modulators (the laser diodes, LDs) at P_1, each imaged onto a separate vertical region of a multi-channel acousto-optic (AO) cell at P_2, with the light leaving P_2 integrated vertically onto a linear CCD shift register detector array with A/D converters and adders on each output detector. This architecture was introduced several years ago [7] and its optical laboratory realization was recently described [6]. Thus, our description and detail of it is brief and our emphasis will be on its laboratory performance and use in the solution of a finite element problem in structural mechanics.

We now briefly describe the use of the system for high-accuracy optical linear algebra operations. Consider one of the M vertical processing channels of the system used to multiply two numbers. The encoded digits of one word are fed sequentially to one of the P_1 point modulator and the digits of the second word are in parallel to the AO cell at P_2. The data for the second word will be present in one vertical section of the AO cell for a time T_2, during which the P_1 point modulator opposite this region of the AO cell is pulsed on each T_1. Each P_1 modulator is pulsed on N times for each word (each T_2) with the N digits of the first word. Thus, $NT_1 = T_2$. Each T_1, one digit of the first word is multiplied by the second word and the output product is formed on the detectors at P_3. The contents of P_3 are then shifted, the next product is formed and added to the prior one, thus accumulating partial products at P_3. After NT_1, the

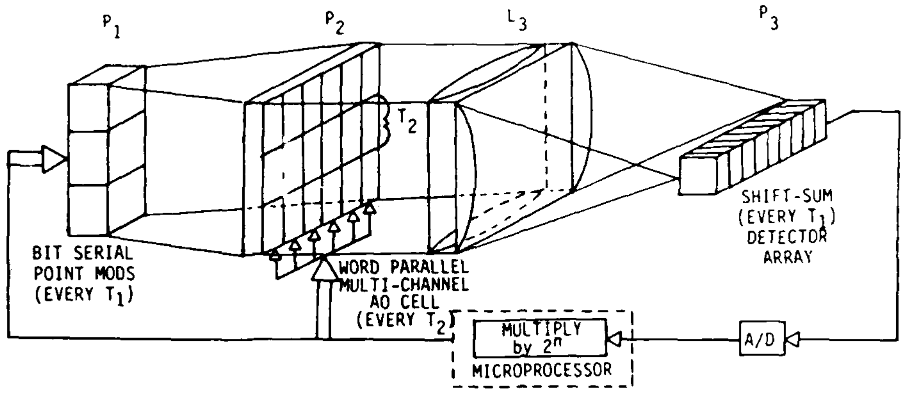

P1
P2
L3
P3
T2
BIT SERIAL POINT MODS (EVERY T1)
WORD PARALLEL MULTI-CHANNEL AO CELL (EVERY T2)
SHIFT-SUM (EVERY T1) DETECTOR ARRAY
MULTIPLY by 2^n
MICROPROCESSOR
A/D

Fig. 1. Multi-channel high accuracy time and space integrating architecture.

P_3 output is the mixed radix representation of the product of the two encoded numbers. These scalar multiplications occur in parallel on the M sections of the system for M different pairs of numbers. Thus, every $T_2 = NT_1$, the system computes an M-element vector inner product (VIP). The mixed radix output can be converted to conventional binary by A/D conversion and with a shift and add of successive digits as they emerge from P_3. This is the digital multiplication by analog convolution (DMAC) [8,9] algorithm for achieving high-accuracy multiplications. This algorithm can be applied to data encoded in any base. However, our present data will use only base-2 encoding.

Many techniques exist to represent bipolar and complex-valued data on such processors [1]. Our present application requires only bipolar data and the algorithm we employ requires only one channel ($M=1$) of the system. Thus, we employ a sign-magnitude negative number representation. In the present laboratory system, the AO cell has an aperture time $T_A = 5$ μs. With $T_2 = 250$ ns and $M=10$, new P_1 data is entered every $T_1 = 25$ ns (a 40 MHz rate per channel). The system allows easy partitioning of problems in which the dimensionality of the matrix exceeds that (M) of the processor. This is achieved by diagonal partitioning as detailed elsewhere [7]. We employ this technique in the data flow on the system by feeding M of the diagonals of the matrix to P_1 and the vector data to P_2 to achieve a matrix-vector multiplication.

In the laboratory system used in this paper, we employed only $N=3$ of the 32 available AO cell channels. This would typically restrict the system's precision to $2^N = 2^3$ or 3 bits. However, the DMAC algorithm allows one to process the different bits of the word separately (since no carries are required until the final mixed radix to binary conversion is done, and this need not be done after each VIP). The problem we consider requires $B=21$ bit accuracy. We achieve this by processing 3 bits of each word each T_2. Thus, we operate the system with $T_2 = (B+N-1)T_1 = 23T_1$ and produce a VIP every $(B+N-1)(B/N)T_1 = 7T_2$, where $T_1 = 0.1$ μs for the laboratory data tests reported upon here. This demonstrates the added flexibility of this system to achieve any desired accuracy by bit partitioning and represents a most unique hardware/accuracy/speed tradeoff possible in this architecture.

Another attractive property of the system of fig. 1 is its ability to easily perform LU decomposition [7]. Our laboratory demonstration employs this algorithm and thus we briefly review its implementation. To solve $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$ by LU decomposition, we decompose the matrix $\mathbf{A}$ into $\mathbf{A} = \mathbf{LU}$ (where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular matrices). This allows us to solve the original problem by back substitution. The decomposition is achieved by multiplying $\mathbf{A}$ successively by N decomposition matrices $\mathbf{P}_m$ (for a matrix $\mathbf{A}$ of size $N \times N$). Synthesis of the decomposition matrix is trivial and requires only the elements of one column of the $\mathbf{P}_m \mathbf{A}_{m-1} = \mathbf{A}_m$ matrix calculated just previously. The structure of each $\mathbf{P}_m$ is quite simple (its diagonal elements are all one and it has non-zero elements in only column m below the diagonal, with all other entries being zero). Thus, the $\mathbf{P}_m\mathbf{A}_{m-1}$ matrix–matrix multiplications required can be obtained using only one channel ($M=1$) of our system as shown in fig. 2 and as we now discuss. The single non-zero off-diagonal elements in each row of $\mathbf{P}_m$ are fed time-sequentially to the single P_1 modulator and the elements of $\mathbf{A}$ are fed word parallel to the AO cell at P_2 and (after a delay) are also fed to the output from the P_3 detector. Thus, taking advantage of the structure of the LU decomposition matrix allows us to use the one channel system of fig. 2 for LU decomposition. In our LU algorithms for the $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$ example, we apply $\mathbf{P}_m$ to the augmented matrix $\mathbf{A}'_m$ which has an additional column with the vector $\boldsymbol{b}$ appended. By applying $\mathbf{P}_m$ to $\mathbf{A}$ and $\boldsymbol{b}$, we produce one row of the matrix $\mathbf{U}$ and one element of the new $\mathbf{U}\boldsymbol{x}=\boldsymbol{b}'=\mathbf{L}^{-1}\boldsymbol{b}=\mathbf{P}\boldsymbol{b}$ vector each T_2, where $\mathbf{P}=\mathbf{P}_1\mathbf{P}_2...\mathbf{P}_m...=\mathbf{L}^{-1}$. These $\mathbf{U}$ and $\boldsymbol{b}'$ elements are then fed to the processor of fig. 1 (or a digital processor) to solve the upper triangular $\mathbf{U}\boldsymbol{x}=\boldsymbol{b}'$ equation for the final $\boldsymbol{x}$ solution by back substitution [10].

3. Finite element case study

The case study chosen for the laboratory system involved the solution of a plate bending problem for the displacements at all nodes with different loading forces and boundary conditions present. This problem was chosen since it was modeled and simulated earlier [11]. We thus only highlight it here, since our present purpose is to solve the problem on an optical laboratory processor. We selected an aluminum plate 6′×8′×1″ divided into 8 rectangular plate bending finite element regions as shown in fig. 3. The structure has $M=15$ nodes with $D=3$ degrees of freedom (displacements, etc.) per node for a total of $N=M\times D=45$ degrees of freedom describing the system. An $N\times N$ stiffness matrix $\mathbf{K}$ describes the system. Optimal node numbering was used to reduce the matrix bandwidth to 29. The boundary conditions involved clamping the top and left edges (the elements denoted by an × in fig. 3). A force was applied in the z direction to the bottom right node (case 1) and to this node and the adjacent edge nodes (case 2). These external loads and forces together with the boundary conditions are described by the N element force vector $\boldsymbol{p}$. The problem is to calculate the three degrees of freedom at the 8 unclamped nodes (24 unknowns), which are described by the N-element vector of displacements $\boldsymbol{d}$. This is achieved by solving the system of linear algebraic equations $\mathbf{K}\boldsymbol{d}=\boldsymbol{p}$ for $\boldsymbol{d}$.

4. Optical laboratory system results

The one-channel system of fig. 2 with three AO cell channels partitioned for 21-bit accuracy was used to solve the $\mathbf{K}\boldsymbol{d}=\boldsymbol{p}$ problem by LU decomposition. In tables 1 and 2 we show the results obtained. Column one lists the 24 internal node degrees of freedom to be calculated. The values obtained on our simulator are listed in column two and the results obtained on the optical laboratory system are given in column three. As seen, all results are identical, thus indicat-

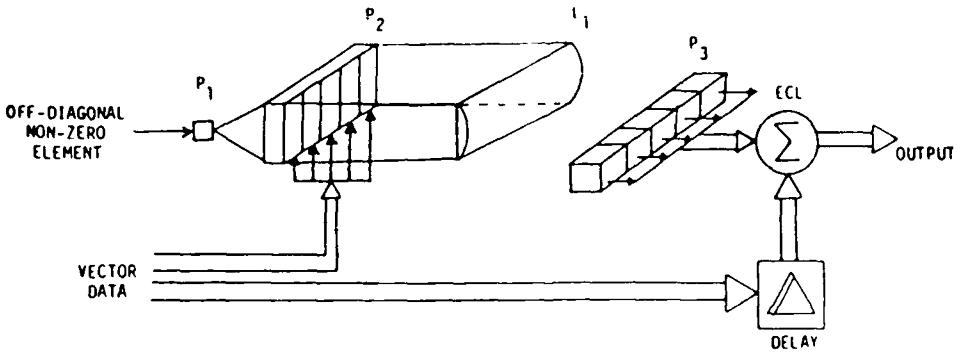


Fig. 2. One-channel LU decomposition architecture for matrix decomposition.

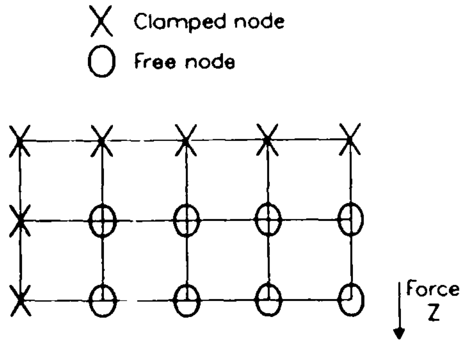


Fig. 3. The aluminum plate finite element structure used for our case study.

ing that the optical processor's results are perfect and that the system produced no errors in all solutions calculated.

Table 1
Simulated and optically calculated values for the 24 unknown displacement vector components for case 1.

Solution vector	Intel simulator results	Optical results
x(0)	25.139	25.139
x(1)	−0.318	−0.318
x(2)	0.460	0.460
x(3)	9.324	9.324
x(4)	−0.123	−0.123
x(5)	0.400	0.400
x(6)	17.459	17.459
x(7)	−0.335	−0.335
x(8)	0.340	0.340
x(9)	5.967	5.967
x(10)	−0.138	−0.138
x(11)	0.280	0.280
x(12)	9.334	9.334
x(13)	−0.318	−0.318
x(14)	0.181	0.181
x(15)	2.974	2.974
x(16)	−0.110	−0.110
x(17)	0.150	0.150
x(18)	2.834	2.834
x(19)	−0.210	−0.210
x(20)	0.040	0.040
x(21)	0.809	0.809
x(22)	−0.066	−0.066
x(23)	0.049	0.049

Table 2
Simulated and optically calculated values for the 24 unknown displacement vector components for case 2.

Solution vector	Intel simulator results	Optical results
y(0)	12.767	12.767
y(1)	−0.204	−0.204
y(2)	0.255	0.255
y(3)	4.140	4.140
y(4)	−0.051	−0.051
y(5)	0.199	0.199
y(6)	8.038	8.038
y(7)	−0.186	−0.186
y(8)	0.158	0.158
y(9)	2.646	2.646
y(10)	−0.064	−0.064
y(11)	0.128	0.128
y(12)	4.020	4.020
y(13)	−0.145	−0.145
y(14)	0.076	0.076
y(15)	1.264	1.264
y(16)	−0.049	−0.049
y(17)	0.065	0.065
y(18)	1.176	1.176
y(19)	−0.089	−0.089
y(20)	0.016	0.016
y(21)	0.328	0.328
y(22)	−0.027	−0.027
y(23)	0.020	0.020

5. Summary and conclusion

The optical laboratory system data described have demonstrated many new points: sign-magnitude negative number representation, partitioning of problems larger than the processor's size, bit-partitioning to increase the accuracy of the system beyond the number of bit channels in the processor, the first direct solution of a system of linear algebraic equations, a new one-channel LU decomposition algorithm, and the first use of optical processors for the solution of finite element problems in structural mechanics.

The laboratory system described used an AO cell (bandwidth BW = 50 MHz) at P_1 and a 10-channel AO cell (BW = 10 MHz) at P_2. Assuming a 20-channel AO cell at P_3, the system performs one 20-bit multiplication in 20/(50 MHz) = 20(0.02) = 0.4 μs or 2.5 MOPS (millions of operations per second). This is quite competitive with personal computer coprocessors (whose 5 MHz clock rate yields 1 MOP performance, typically). With other AO cells, one can easily increase the optical system's bandwidth to 1 GHz (a factor of 20 improvement). With $M=10$ channels at the input plane, we obtain an additional factor of 10 improvement. Thus, a factor of 200 improvement or 2.5(200) = 500 MOP performance is not unrealistic.

Acknowledgement

The support of the laboratory system fabrication by a grant from the Air Force Office of Scientific Research and the use of this system in solving finite element problems by a National Science Foundation grant and partial support from NASA Langley Research Center are gratefully acknowledged.

References

[1] Proc. IEEE, Vol. 72, Special Issue on Optical Computing, July 1984.
[2] Optical Society of America, Meeting on Optical Computing, Tahoe, Nevada, March 1987.
[3] D. Casasent and R. Krishnapuram, Appl. Optics 26 (1987) 3641.
[4] Optical Engineering, Special Issue on Optical Interconnections, Vol. 25, No. 10, October 1986.
[5] D. Casasent, Proc. SPIE 614 (1986) 153.
[6] D. Casasent and S. Riedl, Proc. SPIE 698 (1986).
[7] D. Casasent and B. Taylor, Appl. Optics 24 (1985) 1476.
[8] H.J. Whitehouse and J.M. Speiser, in: Aspects of Signal Processing – Part II, ed. G. Tacconi, NATO Advanced Study Institute, Boston, MA, 1976, p. 669.
[9] D. Psaltis, D. Casasent, D. Neft and M. Carlotto, Proc. SPIE 2332 (1980) 151.
[10] A. Ghosh, D. Casasent and C. Neuman, Appl. Optics 24 (1985) 3883.
[11] B. Taylor and D. Casasent, Appl. Optics 25 (1986) 966.

CHAPTER 8:

OPTICAL LINEAR HETERODYNE MATRIX-VECTOR PROCESSOR

Proc. SPIE
Volume 886
January 1988, O-E/LASE'88

OPTICAL LINEAR HETERODYNE MATRIX-VECTOR PROCESSOR

Eugene Pochapsky and David Casasent
Carnegie Mellon University, Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

An analog optical matrix-vector processor with 10-bit accuracy is described. The operating mode of the various components of the system and the system architecture are reviewed. The system is capable of handling bipolar and complex-valued data with no loss in throughput. Various applications of this architecture and initial laboratory data and simulation results are provided. Applications addressed include: finite impulse response filters, two high/low accuracy algorithms and systems for solving linear algebraic equations, a correlation cancellation loop processor and new algorithms and architectures for the discrete and continuous steepest descent algorithms and solutions, plus preconditioning algorithms and associated techniques for these systems, and finally constrained LAE solutions for reduced accuracy processors (using ridge regression techniques).

1. INTRODUCTION

Optical matrix-vector processors [1] are viable numerical processor architectures. This is especially true if these systems are operated in an analog mode (since this increases their throughput significantly and avoids the need for A/D converters). Such architectures must achieve 8-10 bit accuracy to be viable. Most proposed architectures can achieve only 5-6 bit accuracy and are thus significantly limited in their use. Section 2 reviews the component operating modes required to achieve such performance and provides the analysis of one such optical processor capable of this required performance [2]. Section 3 addresses several applications, algorithms and architectures for this system. These include: finite impulse response (FIR) filters, two high/low accuracy algorithms to improve the results of a linear algebraic equation (LAE) solution and to achieve higher accuracy in an LAE solution, and finally a new correlation cancellation loop analog processor. Section 4 addresses fundamental issues associated with the gradient descent iterative algorithms. These include discrete and new continuous algorithms. The effect of processor accuracy on the problem stability and the associated processor architecture required are also addressed and quantified. Quantitative data that verifies our theoretical analysis is included. Section 5 advances an algorithm to modify an LAE solution to allow reasonable results to be obtained on a processor of limited accuracy. The algorithm requires solution of a minimization problem with a constraint on the accuracy of the processor. The resultant algorithm solution to this constrained problem requires ridge regression techniques to

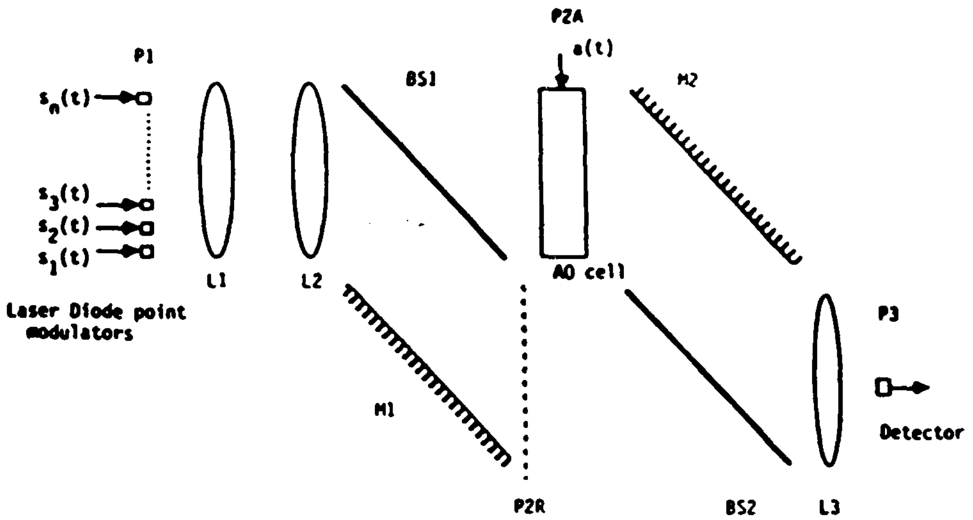


FIGURE 1: Basic optical linear heterodyne optical vector inner product (VIP) processor.

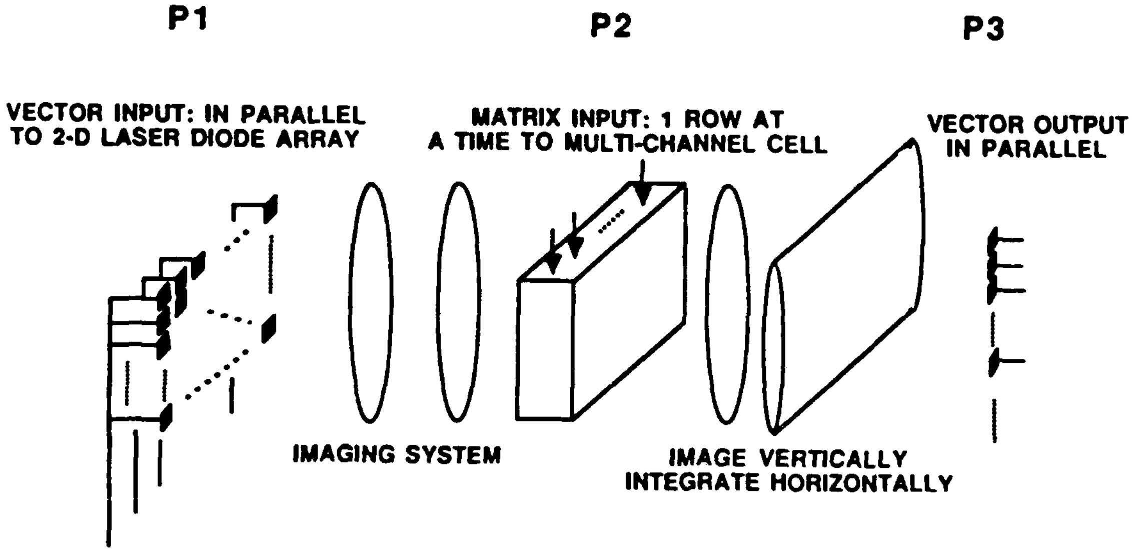

P1
P2
P3
VECTOR INPUT: IN PARALLEL TO 2-D LASER DIODE ARRAY
MATRIX INPUT: 1 ROW AT A TIME TO MULTI-CHANNEL CELL
VECTOR OUTPUT IN PARALLEL
IMAGING SYSTEM
IMAGE VERTICALLY INTEGRATE HORIZONTALLY

FIGURE 2: 2-D matrix-vector version of the system of Figure 1.

no loss in throughput. These represent several very attractive and practical aspects of this system.

3. INITIAL APPLICATION DISCUSSION

The basic space integrating heterodyne processor of Figure 1 functions directly as a FIR filter, as shown in Figure 3. The complex filter weights $w_i(\tau_i)$ are fed to the LDs, the signal to be filtered a(t) is fed to the AO cell, and the filter function

This is the error in the solution. The magnitude of $\underline{r}$ is expected to be on the order of the accuracy of the optical processor (i.e. 0.1% for a 10-bit system). When a more accurate answer is required, we scale $\underline{r}$ by s to form $s\underline{r}_k$ at cycle k through the full high/low accuracy processor. This scaling increases the value of $\underline{r}_k$ at cycle k through the processor. This scaling is necessary to increase $\underline{r}_k$ to allow an analog optical processor to refine and improve the accuracy of the result. After scaling $\underline{r}_k$, we use a low accuracy optical processor to solve the new scaled LAE problem

$$\underline{A}\,\underline{y} = s\underline{r}_k. \tag{5}$$

The refined solution after cycle k+1 is the refined solution with improved accuracy

$$\hat{\underline{x}}_{k+1} = \hat{\underline{x}}_k + \underline{y}/s. \tag{6}$$

The procedure of calculating estimates $\hat{\underline{x}}_k$ of $\underline{x}$ iteratively, refining and scaling the residual error (to high accuracy) and repeating the iterative algorithm (with a scaled vector) on an analog processor to improve the accuracy of the result represents an attractive and viable use of an optical processor to achieve high accuracy LAE solutions.

The next application we consider for an analog optical processor is as a correlation cancellation loop system. The scenario envisioned includes a directional antenna whose output m(t) is the desired signal plus sidelobe jammers, and an omnidirectional adjunct antenna, whose output, which we denote by a(t), includes primarily the jammer signal. The basic correlation cancellation loop algorithm now follows.

An adaptive estimate $\hat{s}(t)$ of the signal in the main channel is calculated from the omnidirectional antenna output a(t), using the system as an FIR filter, as

$$\hat{s}(t) = \sum_i w_i a(t-\tau_i). \tag{7}$$

We then form the difference between m(t) and the estimate $\hat{s}(t)$, i.e. we form the residue

$$r(t) = m(t) - \sum_i w_i a(t-\tau_i). \tag{8}$$

The weights w_i are updated and computed as

$$w_i(t,\tau_i) = \int_0^t a(u-\tau_i) r^*(u) du. \tag{9}$$

A LC filter performs the time integration noted in Eq.(9). Adaptation continues until no correlated noise components remain, i.e. until necessary, because of its higher throughput. The algorithm used to solve $\underline{A}\ \underline{x} = \underline{b}$ is

$$\underline{x}_{k+1} = \underline{x}_k - \omega(\underline{A}\ \underline{x}_k - \underline{b}). \tag{10}$$

We have performed a new accuracy and stability analysis of this algorithm that has related optical system accuracy to APAR (adaptive phased array radar) performance, using signal-to-interference plus noise ratio (SNIR) as the performance measure. Brief remarks are now advanced on this algorithm, architecture and our analysis.

Errors in $\underline{b}$ and $\underline{A}$ affect the accuracy of the result obtained $\underline{x}$. However, errors in the optical representation of $\underline{A}$ (due to optical system component performance and accuracy) are most important as they affect the algorithm's stability (i.e. does the algorithm converge to a useful solution?). The issues to be addressed are: does the algorithm converge?, is the solution obtained useful?, and how fast is the solution obtained? Errors $|e_{ij}|$ in the representation of $\underline{A}$ thus determines stability. Our new analysis shows that stability requires

$$|e_{ij}| < \sqrt{N}\ C(A), \tag{11}$$

where N is the dimensionality of $\underline{A}$ (i.e. the dimensionality of the adaptive array) and $C(\underline{A})$ is the condition number of $\underline{A}$. If $|e_{ij}|$ exceeds the limit in Eq.(11), optical errors can render $\underline{A}$ to be singular, preventing algorithm convergence and producing a meaningless solution. For these reasons, the errors in the representation of $\underline{A}$ are of major concern. Given a processor accuracy of B-bits, i.e. $|e_{ij}| \leq 2^{-B}$, the condition number of $\underline{A}$ is limited to $\sqrt{N}\ 2^B$ for stability reasons. As an example, consider a $B = 9$-bit processor, then this system can solve adaptive array problems with matrices whose condition number satisfies $C \leq 2500$ and with $N = 25$ adaptive elements. This represents a quite useful system for many diverse APAR applications.

To quantify the performance possible, consider the APAR scenario A: a linear phased array with the desired signal or beam direction being 0°, with received noise -10 dB below the desired signal, and with jammers 20 dB above the desired signal at 20° and 30°. We consider the resultant output jammer and signal power versus iteration number for 3 cases: a 32-bit floating point processor (digital accuracy), a 10-bit amplitude processor with 0.01 radian phase errors (typical of the OLHNP system described here) and a 6-bit amplitude processor with no phase errors (typical of the classic optical intensity processor). Figure 5 shows the results obtained. As seen, the 10-bit and floating point results are approximately the same, with the jammer power reduced 18 dB below the signal and with similar convergence for both cases. The 6-bit processor initially adapts, but degrades as iterations continue (due to processor errors) such that the jammer and signal powers are equal after 250 iterations. The APAR scenario B considers many multiple jammers, with jammers 10 dB above the signal at 35°, 40°, 45°, and -10° and jammers 20 dB above the signal at -60° and -70°, plus receiver noise 10 dB below the signal level (0 dB) present also. The data for the 3 processors for this scenario are shown in Figure 6. The 10-bit analog and floating point systems perform comparably (SNIR for both are within 2 dB). The 6-bit processor yields a negative SNIR and its solution is thus

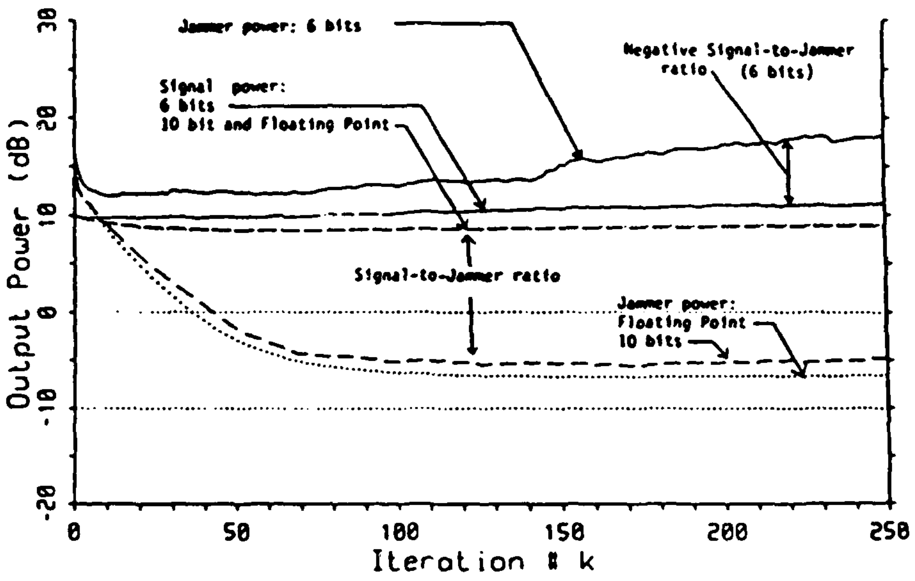


FIGURE 6: Jammer and signal power output vs. iteration number for the multiple-jammer scenario, using 6-bit and 10-bit fixed point, and 32-bit floating point processor accuracy.

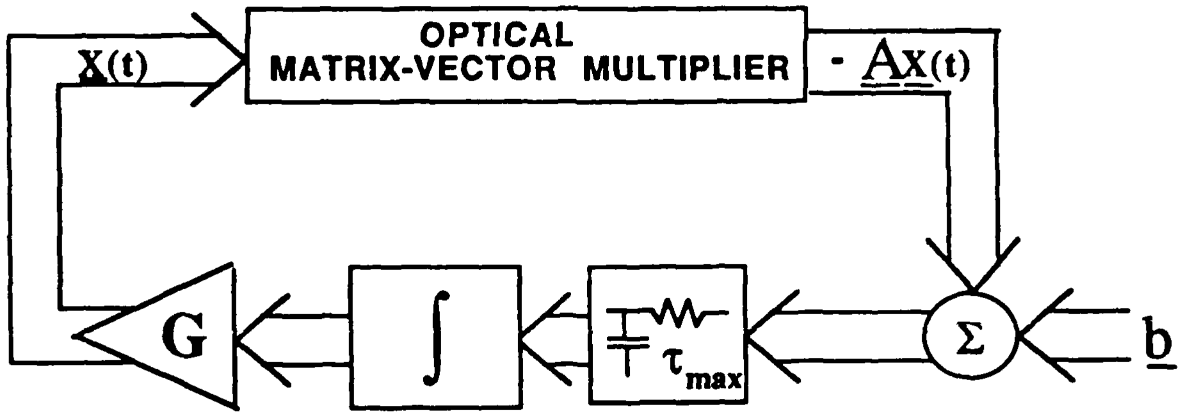


FIGURE 7: Continuous steepest descent algorithm architecture.

5. ALGORITHM LAE SOLUTIONS WITH LIMITED PROCESSOR ACCURACY CONSTRAINTS [3]

We recently [3] devised and tested a new LAE solution algorithm for APAR (suitable for other applications also), in which we maximized the array gain (SNIR) subject to a constraint on the sensitivity of the solution weights $\underline{w}$ to errors in the processor employed. This results in the solution of an LAE in which the diagonal elements of the matrix are perturbed by a constant. This reduces the condition number of the matrix and thus satisfies the accuracy or solution-sensitivity constraint.

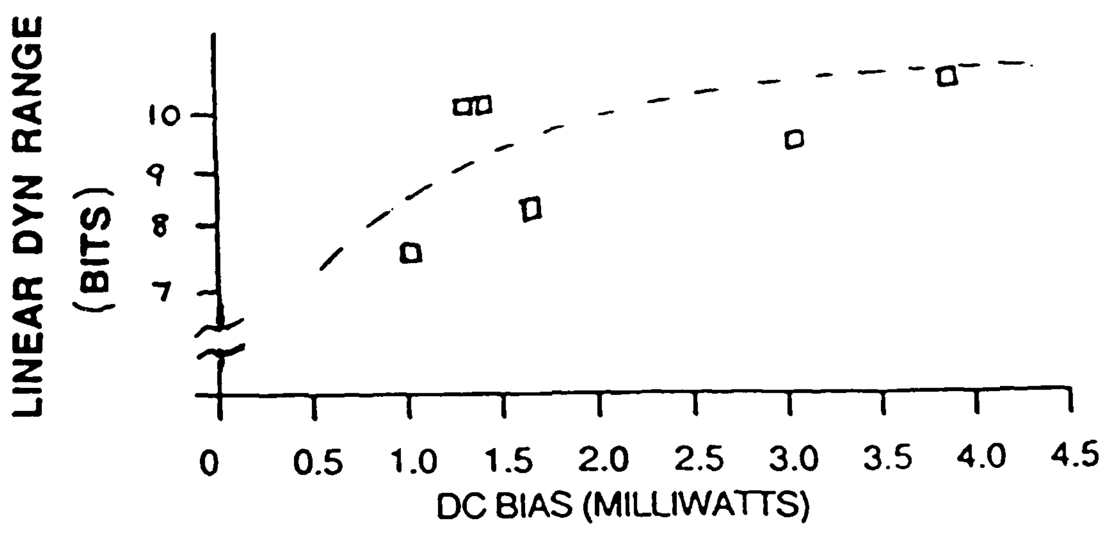


FIGURE 9: Demonstration of laser diode (LD) linear dynamic range and accuracy of 10-bits.

frequencies present (Figure 10) correspond to lines at the dc term (due to local oscillator leakage), the difference frequency (8 MHz), the second harmonic (16 MHz) which is due to detector amplifier nonlinearities (this term can be reduced by higher-performance detector amplifiers) and external RF interference in the laboratory (at 14 MHz). The ratio of the strength of the difference signal at 8 MHz with respect to the background shows a system dynamic range approaching 60 dB to be possible.

The bipolar multiplication ability of this system is demonstrated in Figure 11. The inputs are the bottom 2 traces with polarities indicated, and the output is the upper trace. Figure 12 shows a photograph of the optical laboratory system used.

7. SUMMARY AND CONCLUSION

We have demonstrated that analog optical processors with 10-bit accuracy are possible and feasible and that there is sufficient motivation and need for such systems. The processor described operates the various components of the system in the proper modes for linear operation and temperature stability. The use of heterodyne detection and quadrature modulation provides the system with temperature stability, as well as the ability to handle bipolar/complex data with no loss in system throughput. Various applications and algorithms for use on this system were addressed. These included: FIR filters, correlation cancellation loop processors, discrete steepest descent systems, continuous steepest descent processors, several hybrid low/high accuracy processors, and new constraint algorithm solutions to reduce the condition number of the problem being addressed. Laboratory data was provided to show 60-70 dB performance and 10-bit linear dynamic range possible, as well as bipolar processing with no loss in system throughput.

ACKNOWLEDGMENTS

We acknowledge the partial support of this research by a grant from the Air Force Office of Scientific Research (Grant AFOSR-84-0239) for adaptive phased array radar processing.

REFERENCES

1. Special Issue on *Optical Computing*, Proc. IEEE, Vol. 72, No. 7, July 1984.
2. E. Pochapsky and D. Casasent, "Linear Acousto-Optic Heterodyning Processors for Complex-Valued Data Processing", Proc. SPIE, Vol. 752, pp. 155-171, January 1987.
3. D. Casasent and A. Ghosh, "Reduced Sensitivity Algorithm for Optical Processors Using Constraints and Ridge Regression", Applied Optics, accepted for publication.
4. D. Psaltis, D. Casasent, D. Neft and M. Carlotto, "Accurate Numerical Computation by Optical Convolution", Proc. SPIE, Vol. 232, pp. 151-156, April 1980.
5. H.J. Whitehouse and J.M. Speiser, "Linear Signal Processing Architectures", in Aspects of Signal Processing - Part II, G. Tacconi, ed., NATA Advanced Study Institute, Boston, MA, pp. 669-702, 1976.
6. W. Thompson (Lord Kelvin), "On a Machine for the Solution of Simultaneous Linear Equations", Proc. Royal Society of London., Series A, Vol. 28, No. 191, pp. 111-113, 1878.
7. H.J. Caulfield, J.H. Gruninger, J.E. Ludman, K. Steiglitz, H. Rabitz, J. Gefland, and E. Tsoni, "Bimodal Optical Computers", Applied Optics, Vol. 25, No. 18, pp. 3128-3131, 15 September 1986.
8. M.A.G. Abushagur and H.J. Caulfield, "Speed and Convergence of Bimodal Optical Computers", Optical Engineering, Vol. 26, No. 1, January 1987.

CHAPTER 9:

DISCRETE STEEPEST DESCENT ALGORITHMS AND THEIR REALIZATION ON OPTICAL ANALOG PROCESSORS

Proc. SPIE, Volume 977, August 1988

DISCRETE STEEPEST DESCENT ALGORITHMS AND THEIR REALIZATION ON OPTICAL ANALOG PROCESSORS

Eugene Pochapsky and David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

We consider an analog (linear) heterodyned linear algebraic optical processor for adaptive phased array radar (APAR). Its use in solving the discrete steepest descent (DSD) algorithm is considered. New stability and performance measure expressions are used (that relate the scenario and the processor's accuracy) and their verification is obtained by scenario tests. Extensions to more complex problems by terminating the number of iterations and by matrix preconditioning are discussed and demonstrated.

1. INTRODUCTION

Earlier [1], we advanced an optical linear heterodyned numerical processor (OLHNP) architecture and described the operation of its components and its advantages. These include the use of analog optics (to maintain speed), proper device operation (to achieve accuracy), a new architecture to handle bipolar and complex-valued data (with no loss in throughput), and its ability to correct amplitude, phase and spatial errors (and the necessity for this). Section 2 reviews this system. It can easily be extended from a vector inner product to a 2-D matrix-vector processor [1,2] and to achieve higher accuracy (using encoded data) [1,2]. Its ability to perform 9-10 bit linear algebra operations has been quantified and demonstrated [2] and various applications and extensions of the system have been noted [2]. The application we consider is APAR. This is highlighted in Section 3, where we emphasize the DSD algorithm solution and new theoretical results obtained for the use of an analog processor in a DSD solution. These results are verified with our data in Section 4. Extensions of the algorithm and system are then advanced in Section 4.6.

2. ARCHITECTURE

The basic architecture considered is shown in Figure 1. It uses N laser diodes (LDs) at P_1

imaged onto an acousto-optic (AO) cell at P2A. A reference arm in the system is provided in a Mach-Zehnder architecture. The upper arm of this system provides the sum of the products of the elements of two signals (vectors) $s_n(t)$ and a(t) elements (i.e., it forms the vector inner product (VIP)). The heterodyne architecture shown provides amplitude VIP data output with simple input signals, compatible output signals, plus increased dynamic range and linearity of the system. For high accuracy, the LDs must be operated in an intensity mode with bias B. The input signal to the LD is B+s(t), where s(t) is quadrature modulated. The AO cell is amplitude modulated with its input quadrature modulated. The heterodyne detected P_3 output is the complex-valued VIP of the LD and P2A data [1]. The system of Figure 2 is the matrix-vector version of Figure 1 (the heterodyning section is omitted for simplicity).

3. APAR AND DSD

The basic APAR problem is to solve

$$\underline{R}\,\underline{w} = \underline{s} \tag{1}$$

for the adaptive weights $\underline{w}$ given the noise (antenna or receiver noise) and interference (directional jammers) covariance matrix $\underline{R}$ and the steering vector $\underline{s}$. Iterative algorithms are essential to achieve useful and stable results on an analog processor. The DSD algorithm solution is

$$\underline{w}_{k+1} = \underline{w}_k - \alpha(\underline{R}\,\underline{w}_k - \underline{s}), \tag{2}$$

where k indicates the iteration number and $\alpha = \sqrt{2}/\mathrm{Tr}[\underline{R}]$ is the acceleration parameter used. This acceleration parameter value choice reduces solution errors due to time-varying noise in the OLHNP (and is thus smaller than the conventional value used).

We have analyzed the stability of this algorithm (on an analog processor with B-bit accuracy) and found that errors $|\epsilon_{ij}| \leq 2^{-B}$ in the representation of the elements of $\underline{R}$ require that (for stability)

$$K_t < 2^B, \tag{3}$$

where 2^B is the maximum linear dynamic range of the processor and

$$K_t = (\sum_n \lambda_n) = \lambda_{min} = \mathrm{Tr}[\underline{R}]/\lambda_{min}. \tag{3b}$$

This result is general for any similar iterative algorithm (but does not apply to DMI algorithms). We note that K_t is related to the condition number K_2 of $\underline{R}$. Specifically, $K_t = K_2 + \Sigma\, \lambda_n/\lambda_{min}$

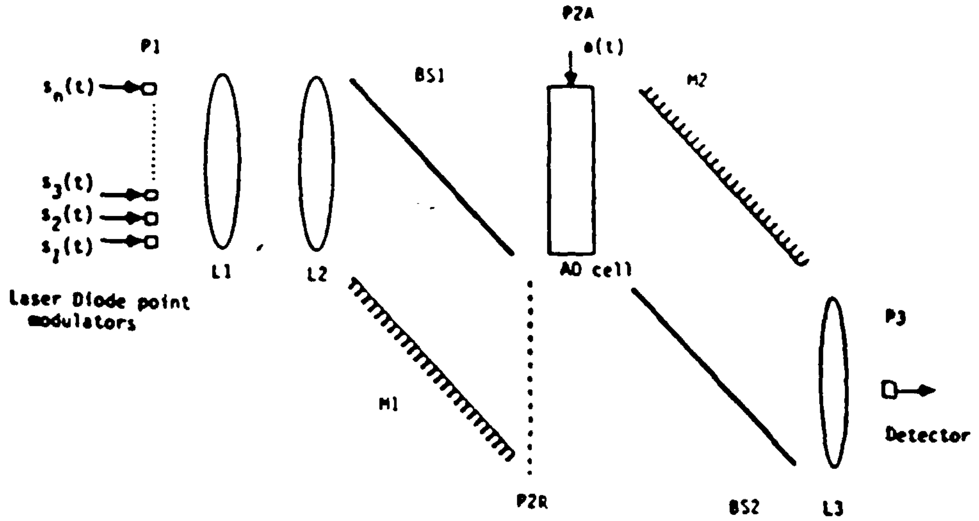


FIGURE 1: Basic optical linear heterodyne optical vector inner product (VIP) processor.

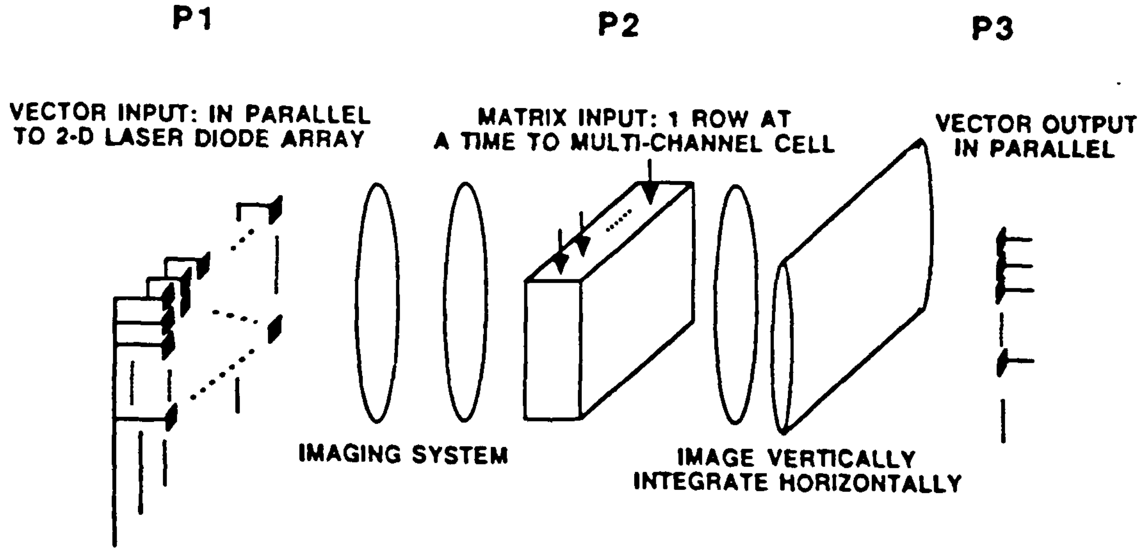

P1
P2
P3
VECTOR INPUT: IN PARALLEL TO 2-D LASER DIODE ARRAY
MATRIX INPUT: 1 ROW AT A TIME TO MULTI-CHANNEL CELL
VECTOR OUTPUT IN PARALLEL
IMAGING SYSTEM
IMAGE VERTICALLY INTEGRATE HORIZONTALLY

FIGURE 2: 2-D matrix-vector version of the system of Figure 1.

(where the sum is over $\lambda \neq \lambda_{max}$) and K_t has a maximum value of $(N-1)K_2$, where N is the number of adaptive elements in the array. One can obtain (3) by considering the effect of accuracy (B) on the positive-definite (positive eigenvector) property of $\underline{R}$. Specifically, we recall that with one dominant eigenvector (or jammer), $K_2 \approx K_t$. We also recall that the dominant eigenvectors are associated with the jammers and the minimum eigenvectors with the antenna noise. We also recall that as new jammers appear, more large eigenvectors occur and that $\underline{R}$ must be normalized by multiplying its elements by $N/\mathrm{Tr}[\underline{R}]$, so that $\underline{R}$ can be represented on the processor. Since K_2 does not change as strong jammers (with approximately the same λ_{max}) are added, the change in K_t due to $N/\mathrm{Tr}[\underline{R}]$ scaling reflects the harder problem being solved and the increased effect that the accuracy of the processor (B) will have on the result.

Golub [3] notes that the normalized root MSE is less than K_2 times N times the error element in the error matrix (this is consistent with our analysis and use of K_t rather than the condition number K_2.

Thus, the condition number K_2 alone does not describe the effect of processor accuracy on performance, since the scaling of $\underline{R}$ results in smaller elements (components) in $\underline{R}$. The smaller components of $\underline{R}$ (in an eigenvector decomposition) correspond to the antenna noise and the larger components correspond to the jammers (with large eigenvectors). Thus, we expect a low accuracy processor to still achieve good jammer nulling (large eigenvectors) and its error effects to be dominated by antenna noise. As the preferable global performance measure, we use SNIR (signal to noise plus interference ratio). The SNIR obtained on a converged processor (with B-bit accuracy) for a scenario described by a correlation coefficient p (where 1-p is the correlation between $\underline{s}$ and the directions of the eigenvectors associated with the jammers; thus, p is small for correlated signal and jammers, as arises in the case of many jammers or in the case of mainbeam jammers) and by L (the number of eigenvectors with $\lambda_i \approx \lambda_{min}$ is related to the optimum SNIR (obtained with a Wiener solution of (1) for $B \rightarrow \infty$) by

$$E\{SNIR\} \approx SNIR_{opt}\left[\frac{1 + K_t^2/3p \cdot 2^{2B}}{1 + L\,K_t^2/3p \cdot 2^{2B}}\right]. \tag{4}$$

The errors in the direction of the eigenvectors associated with the jammers (large eigenvectors) are small and the maximum errors occur in the direction of the small eigenvectors. Thus, as L increases, more directions are allowed for noise and performance degrades (since more eigenvectors have minimum or small eigenvalues, where errors are the most). As the accuracy of the processor (B) decreases, so does performance, and as the correlation of the size and jammers (p) increases, performance also degrades (since the problem being solved becomes harder). Thus, (4) follows and in such cases, we expect lower optimum SNIR and that the SNIR obtained with our processor will diverge further from the optimum.

For the SNIR performance measure, we can note several regions and the performance expected in each. If

$$K_t^2 \ll 2^{2B} \cdot 3p/L, \tag{5}$$

then $E\{SNIR\} \simeq SNIR_{opt}$ and there is no loss in performance. If

$$K_t^2 \geq 2^{2B} \cdot 3p/L, \tag{6}$$

then *log*[E{SNIR}] decreases linearly as K_t^2 increases. We also note that the worst-case is

$$\text{Worst Case } E\{SNIR\} = (1/L)SNIR_{opt} \tag{7}$$

which is not necessarily a significant degradation as the maximum of L is N. For guaranteed stability, we must satisfy (3), i.e.

$$K_t^2 < 2^{2B}. \tag{8}$$

We note that useful solutions still result when (8) is not satisfied, i.e. solutions under unstable conditions will be useful if we limit the number of iterations used. This is possible and the choice of the number of iterations required is independent of the scenario and depends only on the processor accuracy. This occurs since the rate of growth of the errors is not a function of the actual eigenvectors. For the B=10 bit analog processor we consider, we employ 100 iterations (we note that the errors build at a known rate proportional to the amount of error in the processor (which follows since the errors in the smallest eigenvector will be proportional to the inverse of the accuracy of the processor). In cases when we cannot terminate the number of iterations (i.e. when $\underline{R}$ and the scenario change more rapidly than the solution obtained), we must satisfy the stability condition in (8). To guarantee stability, the preconditioing in Section 4.6 must be used. Otherwise, the 10-bit processor should always be terminated after 100 iterations. This requires that we restart the processor and begin to calculate a new weight vector. This is not attractive for dynamic $\underline{R}$ scenarios (Section 4.6).

A typical range of p value is 0.1 to a maximum of 1.0. Thus, we see how L, p and B affect our performance measure. We note that stability is <u>guaranteed</u> for $K_t < 2^B$, but that useful solutions are obtained for $K_t > 2^B$ (if we terminate the iterations while the noise and interference is low and before the optical processor errors accumulate). It is possible to calculate the number of iterations at which to terminate the DSD algorithm (independent of the scenario). For B = 10 bits, we find that 100 iterations is a reasonable termination point. We note that at this number of iterations that the SNIR obtained will be close to the optimum when $K_t < 2^B$, and that beyond 100 iterations we approach the asymptotic limit of E{SNIR} in (4). The processor's MSE is thus proportional to $(1/\lambda_i^2)$, where the square arises because of the MSE parameter used (and thus the number (L) of eigenvectors with $\lambda_i \simeq \lambda_{min}$ affects performance). We also note that the worst case SNIR is only 1/L (a maximum of 1/(N-1)) of $SNIR_{opt}$, which can generally be acceptable. Finally, we note that even if the stability condition in (3) is not satisfied, we can still obtain a meaningful solution (if we terminate the iterations). This occurs since the optimum solution diverges slowly (with iterations on new input data) as long as the jammer scenario does not change rapidly (i.e. if we can terminate the DSD iterations before the effect of antenna noise or changing scenario increases).

4. SIMULATION RESULTS

The effects of spatial gain errors are not of concern in our processor (due to correction). Thus, we consider only additive time-varying P_1, P_2 and P_3 errors and nonlinear P_1-P_3 electronic fixed errors.

4.1 ANTENNA SCENARIO

We consider a linear array of N=8 elements with spacings $\lambda/2$. This yields the antenna pattern shown in Figure 3 (for a boresight steering vector). Its mainbeam width is 10°. In the scenarios we consider, all sources and jammers are referenced to a 0 dB antenna noise level. For this antenna, we note that

$$\text{Signal Gain} = 10\log N^2 = 10\log 64 = 18 \text{ dB}. \tag{9}$$

We use 10*log*, since an increase of N^2 in signal power is obtained from an N-element array. Thus, the expected (and observed) signal power is 18 dB above the signal level. In the scenarios we consider, the strength and number of jammers (and their location) are varied.

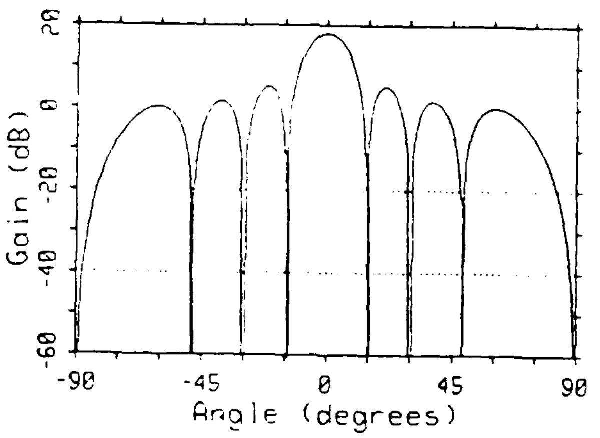


FIGURE 3: Antenna pattern.

4.2 STABLE SOLUTION (SCENARIO-1)

As scenario-1, we consider a single strong jammer (+20 dB at 25°) with a source of +10 dB at 0°. This results in a scenario with L = 7 (with one jammer, there is one large eigenvector and the seven others are small) and p = 0.6 (this is a correlation 1-p = 0.4 that is the mid-range value of the correlation between the signal and the eigenvectors of the jammers). This occurs because the jammer is located near the peak of the first sidelobe of the unadapted antenna pattern, else p would be larger. This scenario is also characterized by $K_2 = 800$ and $K_t = 808$ (these values are close since there is only one jammer eigenvector and this determines λ_{max}). The condition in (8) is satisfied (since $2^{2B} \cdot 3p/L = 270{,}000 < K_t^2 = 650{,}000$). Thus, we expect SNIR close to the optimum (19 dB, calculated from the $B \rightarrow \infty$ Wiener solution as the signal output (28 dB) minus the noise plus interference output of 9dB, i.e. 28-9 = 19 dB = $SNIR_{opt}$). We note that SNIR $\simeq$ $SNIR_{opt}$ at 100 iterations (where we terminate) and that this occurs since the jammer is nulled very rapidly (4 iterations) before processor error effects start to dominate (at $\simeq$ 300 iterations, as seen from Figure 4 and as can be predicted by theory). From (4), we find the asymptotic SNIR, $SNIR_{asy} \simeq$ 14 dB at 100 iterations (in Figure 4), in agreement with the theoretical value calculated from (4). For uncorrelated signals and jammers, $SNIR_{opt}$ will equal the processing gain (18 dB).

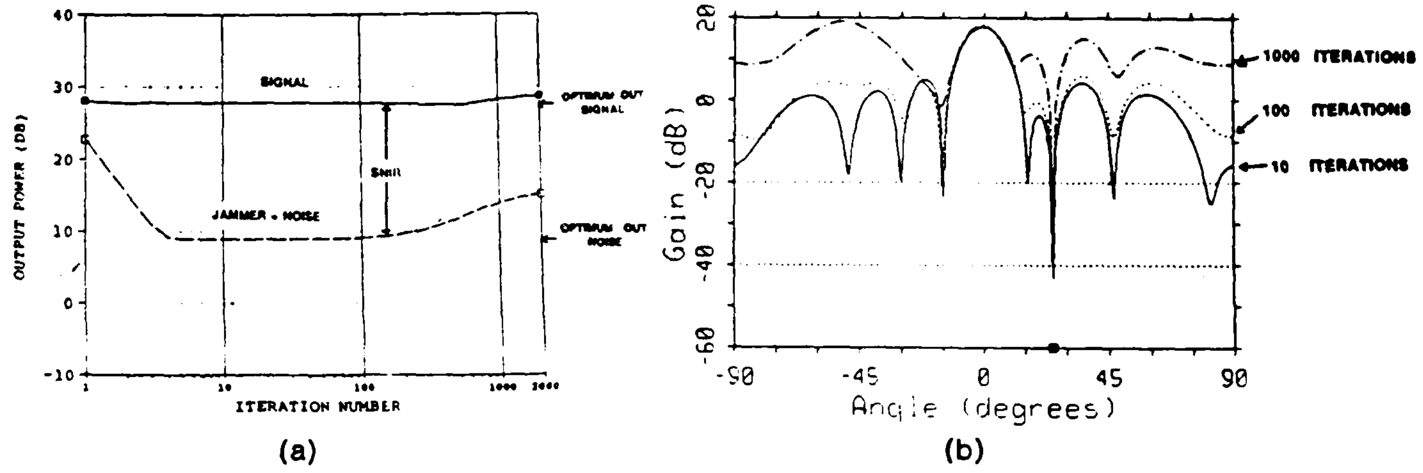


FIGURE 4: (a) Power vs. iteration number k and (b) Antenna pattern obtained after k = 10, 100 and 1000 iterations for scenario-1 (one strong jammer)

Figure 4a shows the signal, jammer and noise, and SNIR obtained. The optimum signal and noise plus interference levels are indicated in the right of the figure. The antenna pattern at different iterations (Figure 4b) shows a worse SNIR at 1000 iterations than at 100, but that the jammer is still well-nulled. This indicates why we use SNIR as the preferable performance measure rather than jammer null depth (which does not reflect noise suppression and signal effects). If the iterations were allowed to continue, the slight rise in signal power due to processor errors will become larger until processor errors have accumulated to the maximum value (> 2000 iterations).

4.2 SNIR DEGRADATION (SCENARIO-2)

Our second scenario involves a mainbeam jammer (20 dB at 5°). For this case p = 0.18

(highly correlated), L = 7 (many jammers and eigenvalues near λ_{min}), K_t = 808 and K_2 = 800. In this case, $K_t^2 \gg 2^{2B} \cdot 3p/L$ and thus from (6), we expect significant degradation in E{SNIR} in (7), i.e. (4) predicts an E{SNIR} of 8 dB or 6 dB less than the optimum SNIR of 14dB. This agrees well (8 dB versus 10 dB) with the value obtained after 1000 iterations. We note that the value obtained at 100 iterations is the optimum SNIR (14 dB). These results follow as expected, since a strong jammer corresponds to a large eigenvelue and will also have a large gain (as shown in Figure 5 at k=1) since this is a mainlobe jammer. Thus, we expect slightly more iterations than in scenario 1 (k=6 in Figure 5) to null this jammer. The jammer related eigenvalue is large and produces the ringing seen in the solution for the signal strength. As before, when we terminate at k=100 iterations, good performance (near the optimum) is obtained.

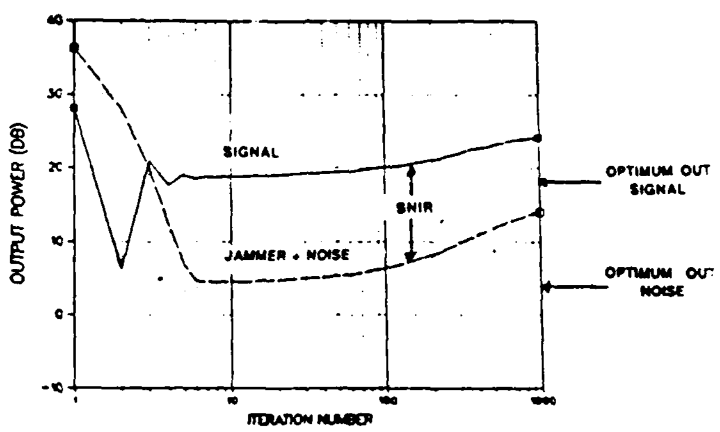


FIGURE 5: Power versus iteration number k for scenario-2 (strong mainbeam jammer, degraded SNIR).

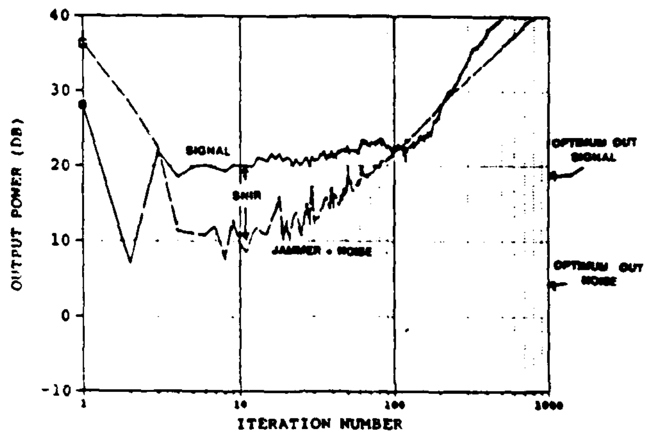


FIGURE 6: Power vs. iteration number k for scenario-2 (Figure 5) using a 6-bit processor.

4.3 LOW ACCURACY PROCESSOR (B = 6 vs. 10) INSUFFICIENT (SCENARIO-2)

Here (Figure 6) we rerun scenario-2 (Figure 5) using a lower accuracy (B = 6 vs. 10-bit) processor. The results show that solution errors rapidly accumulate (with jammer and noise error effects starting to increase at only 10 iterations). Thus, when operating with a reduced accuracy processor, we must terminate the iterations at k = 10 (this number is predictable, independent of the scenario) to achieve useful results (with $\simeq$ 5 dB less SNIR than one can obtain with a 10-bit processor). As shown, higher (10 versus 6 bit) accuracy processors yield considerably better performance.

4.4 MULTIPLE JAMMERS (SCENARIO-3)

This scenario considers 5 jammers (10 dB at 10°, 16 dB at 20°, 10 dB at 30°, 16 dB at

-45°, and 10 dB at -70°). This scenario results in p = 0.4, L = 3, and thus K_2 = 400 and K_t = 880. As in scenario-1, these signals and jammers are fairly correlated (p = 0.4), but L is small (L = 3, since the 5 jammers correspond to 5 of the 8 eigenvectors being large). The $SNIR_{opt} \simeq$ 15 dB. Since L is small, we expect little degradation and from (4) we find E{SNIR} = 13 dB. This scenario data (Figure 7) shows SNIR = 15 dB $\simeq$ $SNIR_{opt}$ at 100 iterations (again this gives nearly optimum performance) and $SNIR_{asy}$ = 12 dB at 1000 iterations (this is within 1 dB of the predicted asymptotic SNIR).

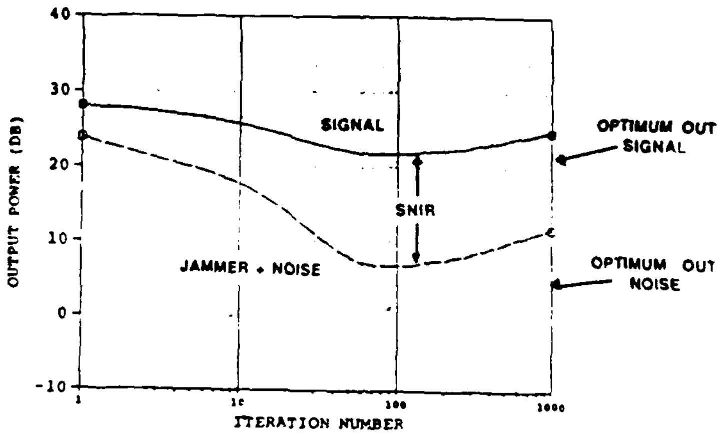


FIGURE 7: Power vs. iteration number k for the multiple jammer scenario-3.

4.5 UNSTABLE SOLUTION (SCENARIO-4)

This scenario considers two very strong jammers (+30 dB at 10° and 40 dB at -30°). The presence of a very strong jammer causes a large condition number (K_t = 88,000). This solution is thus not guaranteed to be stable on our B = 10-bit system. The optimum SNIR (with $B \rightarrow \infty$) possible is 32 dB. In the data obtained, we find a very good SNIR = 40-14 = 26 dB (but not the maximum $SNIR_{opt}$ = 32 dB possible). In this case, the antenna noise is now below the jammer level at 100 iterations and thus does not appear in Figure 8 (and thus the optimum SNIR is not obtained, since the jammer is not sufficiently suppressed). The effect of processor accuracy on stability is not apparent until 500 iterations. The increased noise in Figure 8 after 500 iterations, is due to the low eigenvectors associated with the antenna noise. The processor accuracy increases both the output antenna noise and the output jammer noise. The major effect is due to the antenna noise (since it is associated with the small eigenvectors) and thus the effect of processor errors does not appear until 500 iterations (since the antenna noise is now below the jammer noise). These data show that useful results are still obtained for a problem that is unstable with the given processor accuracy (B) if we terminate the iterations. The SNIR obtained is not predicted by (4), but can be obtained from a new equation (without the asymptotic approximation which results in Eq.(4) when $K_t \leq 2^B$).

4.6 PRECONDITIONING PERFORMANCE

One can improve the asymptotic performance by controlling the condition number of the problem. Two techniques to achieve this are ridge regression [4] and matrix inversion estimation [5-7]. Ridge regression involves adding a value B to the diagonals of $\underline{R}$. This value can be automatically determined [4]. However, although processor errors are reduced, the optimal SNIR value obtained is scenario-dependent. It increases all eigenvalues by B, with negligible effect on large and medium eigenvectors (and thus does not affect the suppression of the jammers they correspond to). It increases the minimum eigenvectors and thus reduces the condition number of $\underline{R}$ and thus the difficulty of the problem. This changes the problem being solved, suppressing components due to the smaller eigenvalues $\underline{R}$ more than they should be, but also significantly reduces processor errors associated with the same small eigenvalues. Thus, with this technique, we can expect better SNIR even after a large number of iterations. Our data in Figure 9 confirms this prediction. In this case, we added 1/500 to the diagonals of the scaled $\underline{R}$ in scenario-4 (the value was automatically determined by the processor accuracy). This reduced K_t from 88,000 to 500. The solution was now stable on a 10-bit processor. In the results obtained (Figure 9), we find no increase in noise and jammer strength at 1000 iterations (this shows a stable solution) but a slightly lower 12 versus 14 dB SNIR at 100 iterations than before preconditioning (Figure 8). Preconditioning allows one to use more iterations. This is preferable in a changing scenario (where one does not want to use a fixed number of iterations, stop and then restart the processor from zero). By preconditioning, (Tr($\underline{R}$)/N)/500 has been added to all eigenvectors. This changes the scenario solved. It gives negligible increase to the strong jammers and thus has negligible effect on their suppression. It increases small eigenvectors (associated with antenna noise) and thus provides a stable solution (with less SNIR than the optimum, with infinite accuracy).

Another approach to preconditioning is to obtain an estimate $\underline{R}^{-1}$ of the inverse by performing a limited number of iterations of the steepest descent algorithm for the LAEs $\underline{R}\ \underline{X} = \underline{I}$, with final solution $\underline{X} = \underline{R}^{-1}$. Premultiplying both sides of (1) by $\hat{R}^{-1}$, where $\underline{R}^{-1}$ is the estimate obtained of $\underline{R}^{-1}$, reduces [5-7] the condition number of $\underline{R}$ to that of $\hat{\underline{R}}^{-1}\underline{R}$. The major problem is that new updates of $\underline{R}$ also have to be similarly preconditioned. Thus, the first preconditioning algorithm is preferable.

5. SUMMARY AND CONCLUSION

The usefulness of a linear 10-bit optimum vector inner product or matrix-vector processor solution of linear algebraic equations for adaptive phased array radar has been quantified. A new analysis of the effects of processor accuracy on stability was noted and proven by examples. A new SNIR performance measure equation (that includes the effects of accuracy and scenario) relating the SNIR obtained to the optimum SNIR, was obtained and shown. Useful solutions were shown to be obtainable using two distinct operating modes. The first mode runs the processor for a fixed number of iterations which is a function of processor accuracy only.

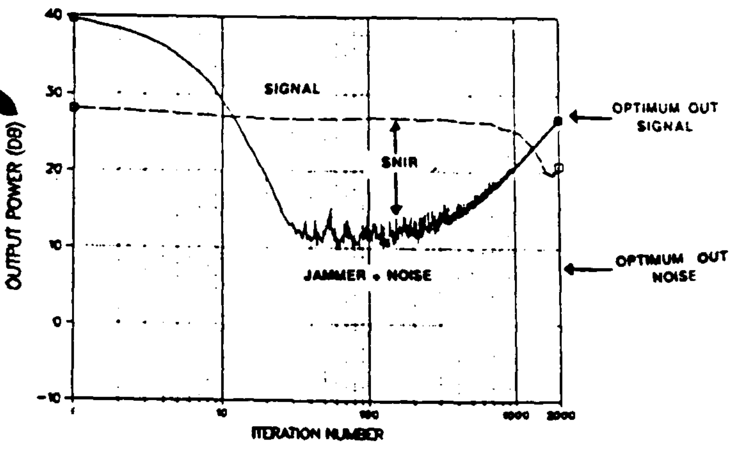


FIGURE 8: Power vs. iteration number k for an unstable problem (scenario-4) showing useful results if the iterations are properly terminated.

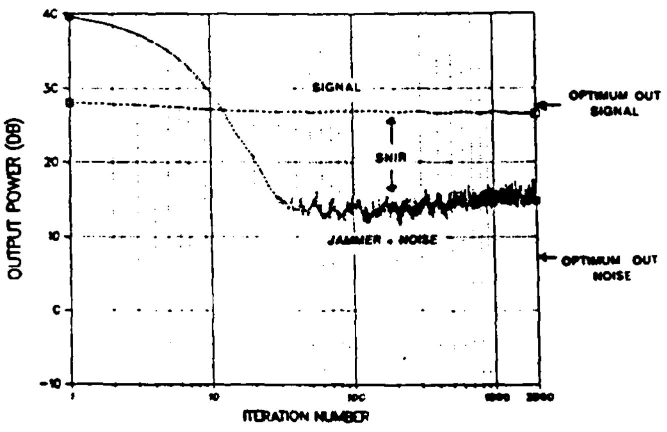


FIGURE 9: Power vs. iteration number for the preconditioned scenario-4.

This mode results in the highest output SNIR but requires restarting the processor after termination. The second mode uses ridge-regression preconditioning to prevent processor errors from accumulating after a large number of iterations, allowing the processor to accommodate a rapidly changing environment at the cost of a small reduction in SNIR. The amount of preconditioning is a function of processor accuracy and the scenario. Extensions to solve ill-conditioned problems by terminating the number of iterations and by ridge regression were described and demonstrated for a wide range of scenarios. In general, we would apply preconditioning in all cases (since it helps far more than it can hurt). If the nearly optimum SNIR is desired (and the jammer scenario does not change rapidly), then we terminate the iterations at a number determined by the accuracy (B-bits) of the processor, and restart at $K = 0$ to calculate the next set of weights.

ACKNOWLEDGMENT

The support of this research by a grant from the Air Force Office of Scientific Research (Grant AFOSR-84-0239) is gratefully acknowledged.

REFERENCES

1. E. Pochapsky and D. Casasent, "Linear Acousto-Optic Heterodyning Processors for Complex-Valued Data Processing", Proc. SPIE, Vol. 752, pp. 155-171, January 1987.

2. E. Pochapsky and D. Casasent, "Optical Linear Heterodyne Matrix-Vector Processor", Proc. SPIE, Vol. 886, January 1988.

3. G.H. Golub and C.F. VanLoan, Matrix Computations, Johns Hopkins Univ. Press,

Baltimore, Maryland, 1983.

4. D. Casasent and A. Ghosh, "Reduced Sensitivity Algorithm for Optical Processors Using Constraints and Ridge Regression", Applied Optics, Vol. 27, pp. 1607-1611, 15 April 1988.

5. A. Ghosh, 'Matrix Pre-Conditioning: A "Robust" Operation for Optical Linear Algebra Processors', Applied Optics, Vol. 26, No. 14, pp. 2734-2737, 15 July 1987.

6. A. Ghosh, "Realization of Preconditioned Lanczos and Conjugate Gradient Algorithms on Optical Linear Algebra Processors", Applied Optics, Vol. 27, No. 15, pp. 3142-3148, 1 August 1988.

7. A. Ghosh, "Performance Analysis of Matrix Preconditioning Algorithms on Parallel Optical Processors", Proc. SPIE, Vol. 939, pp. 19-28, April 1988.

CHAPTER 10:

PATTERN SYNTHESIS USING FOURIER TRANSFORMS

Submitted to Optical Engineering on 10/05/88

Pattern Synthesis Using Fourier Transforms

C.W.Carroll
Mellon Institute, Computer Engineering Center

and

B.V.K.Vijaya Kumar
Electrical and Computer Engineering Department

Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

Many Adaptive Phased Array Radar (APAR) high-resolution spectral estimation techniques are used to determine farfield signal power and location. Once this information is known, placing nulls at these locations to cancel jammers can be accomplished through a proper choice of antenna weights. The antenna weight and angular pattern domains are related through Fourier transformation. To obtain a fine sampling in the angular domain to accurately specify the desired nulls, it is required to extend the antenna aperture by padding it with zeros. However, in the final weight vector applied to the antenna output, the contribution of these extra elements must be zero since they do not correspond to available antenna elements. This provides two sets of constraints on the solution, the set of desired nulls in the angular domain and the available aperture in the weight domain. A method of finding a solution which matches constraints in both the time and frequency domains is the Gerchberg-Saxton algorithm, which is often applied to image reconstruction. This paper will describe the investigation into the behavior of this algorithm as applied to the discrete antenna pattern synthesis case. The algorithm is presented in matrix/vector form and its transient and steady state response is derived. To assist in this analysis, we introduce a new matrix operator which greatly simplifies the required derivations. Computer simulation and numerical evaluations of the analytical results are included to demonstrate the applicability of the algorithm to pattern synthesis.

1. INTRODUCTION

The general phased array radar signal environment is assumed to consist of I narrowband sources in the farfield of an antenna with radar wavelength λ. The K antenna elements are assumed to be equally spaced along a line, separated by a distance d, with identical, unit, isotropic responses. The physical center of the array is defined as the zero time lead/lag point, or the zero phase reference point. This array geometry is illustrated in Figure 1. The signals from the antenna elements are comprised of random antenna noise $\eta_k(n)$ and the sum of the phase shifted signals $p_i(t)$ from the I narrowband sources in the antenna farfield[1]:

$$(\mathbf{x})_{k1}(n) = \eta_k(n) + \sum_{i=1}^{I} p_i(nT)\exp(j2\pi \sin(\Theta_i)\frac{d}{\lambda}(k-1)), \tag{1}$$

where T is the discrete sampling interval and Θ_i is the angle of arrival of the ith signal. We denote the i,jth element of a matrix $\mathbf{x}$ as $(\mathbf{x})_{ij}$ and generalize a vector as a matrix with column dimension of 1. Equation 1 assumes that the zero phase reference point of the array corresponds to the $k=1$ antenna element. All angles are specified relative to the array perpendicular.

For a single narrowband signal with instantaneous amplitude P and no noise, the received signal at the kth antenna element can be written as

$$(\mathbf{x})_{k1} = P\ \exp(j2\pi\sin(\Theta)\frac{d}{\lambda}(k-1)), \tag{2}$$

where $1 \leq k \leq K$. The array output is the weighted sum of the received signals given by

$$y = \mathbf{w}^H\mathbf{x} \tag{3}$$

where $\mathbf{w}$ is the weight vector and ^{H} denotes the Hermitian or conjugate transpose. Substituting equation 2 into equation 3 yields

$$y = P \sum_{k=1}^{K} (w^*)_{k1} \exp(j\frac{2\pi}{N}((1+\sin(\Theta)\frac{d}{\lambda}N)-1)(k-1)). \tag{4}$$

Let us define a discrete direction q in terms of the continuous angle Θ as

$$q = 1+(\mathrm{ROUND}(\sin(\Theta)\frac{d}{\lambda}N)\ \mathrm{MOD}\ N). \tag{5}$$

The function ROUND() indicates a rounding of the parameter to the nearest integer value. The operator (x MOD N) specifies the non-negative integer remainder obtained by dividing x by N. The range of q is thus $1 \leq q \leq N$. These bounds on the value of q are assumed to be in force throughout this paper unless explicitly stated otherwise. The complex conjugate of the array gain in direction q can now be written as

$$(g^*)_{q1} = (\frac{y}{P})^* = \sum_{k=1}^{K} (w)_{k1} \exp(-j\frac{2\pi}{N}(q-1)(k-1)). \tag{6}$$

Equation 6 indicates that for narrowband signals, the complex conjugate of the angular distribution of the gain **g** and weight **w** are related through a Discrete Fourier Transform (DFT). The time variable in the DFT is equivalent to the antenna element number k shifted by a constant. The variable q represents the discrete frequency variable in the DFT. The dependence of the frequency variable on $\sin\Theta$ demonstrates that a uniform sampling in q *implies a nonuniform sampling in* Θ. The use of the complex conjugate of **g** is merely a notational convenience which allows manipulation of the DFT and **w** instead of the Inverse DFT and $\mathbf{w}^*$.

To place antenna response nulls in the jammer directions, we specify $(\mathbf{g})_{q1}=0$ at the q values corresponding to the jammer angles. The Inverse Fourier transform of **g** then yields the weights **w**. However, due to the limit in the total number of antenna elements, K, the angular resolution in q is poor. To improve this sampling, we can pad **w** with zeros so that its new length is N and then take the transform. In the final weight vector applied to the antenna output, the contribution of these extra elements must be zero since they do not correspond to available antenna elements. This provides two sets of constraints:

$$(\mathbf{g})_{q1}=0 \text{ for } q \text{ corresponding to each jammer location}, \tag{7}$$

$$(\mathbf{w})_{k1}=0 \text{ for } k>K. \tag{8}$$

The jammer location constraints in equation 7 can be written in terms of a set $C=\{c_1,c_2, \ . \ . \ . \ ,c_{max}\}$ containing the values of q for which the gain is desired to be zero.

A method of finding a solution to match constraints in both the time and frequency domains is a generalization of the Gerchberg-Saxton algorithm[2]. It is known as the Error Reduction algorithm and Fienup[3] has provided a summary of its use in the retrieval of phase information from intensity data. It has also been used for other applications where constraints can be formulated in both domains such as amplitude reconstruction from phase information[4], bandlimited signal and image extrapolation[5, 6], and the design of spatial filters that provide invariance to 3-D distortions[7]. The algorithm involves repeated FT's where the resulting functions are forced to meet the constraints first in one domain and then in the other. A block diagram of this algorithm is presented in Figure 2. Starting with the weight vector $\mathbf{w}$, it is first transformed into the angular gain domain by the DFT block. The angular constraints are then applied to the gain vector followed by the inverse DFT to return to the weight domain. Finally, the antenna constraints are imposed to complete one iteration of the algorithm. The matrices performing the operations in each block are described more fully in Section 3. The repetitive use of the Fourier transform operation in this algorithm is well-suited to implementation on an optical system. The optical system previously proposed for use in the extrapolation of bandlimited images[8] can be used here. In the paper, we will focus our attention only on the algorithm itself and not its implementation.

The remainder of this paper describes the investigation into the behavior of this algorithm as applied to the discrete case. We first set up the algorithm in matrix/vector form in Section 2. Under specific investigation are the values of the weights (Section 3) and the angular domain constraint error (Section 4) as functions of the number of iterations of this algorithm. To assist in this analysis, we introduce a *new* matrix operator which greatly simplifies the required derivations. Computer simulation and numerical evaluations of the analytical results for sample APAR test scenarios are described in Section 5.

2. MATRIX/VECTOR FORMULATION

In this section, we first set up the iterative step shown in Figure 2 in matrix/vector form. Each of the four functional blocks can be implemented through the pre-multiplication of a vector by a matrix. The resulting vector is then passed to the next block in the cascade. In describing the progression, we begin with the $K\times 1$ weight vector $\mathbf{w}$. The zero padded DFT is obtained in two steps. The vector is first extended to length N through pre-multiplication by the matrix $\mathbf{T}_K^T$, where $\mathbf{T}_K$ is a $K\times N$ truncation matrix whose first K columns form a $K\times K$ identity matrix and with zeros in the remaining columns. This is followed by a pre-multiplication by the $N\times N$ DFT matrix $\mathbf{D}_N$, which transforms the vector into the angular domain. The elements of this matrix are defined as

$$(\mathbf{D}_N)_{ik} = \exp(-j\frac{2\pi}{N}(i-1)(k-1)) \qquad 1 \leq i,k \leq N. \qquad (9)$$

It should be noted that this DFT matrix is symmetric and that its inverse is given by $\mathbf{D}_N^{-1}=\frac{1}{N}\mathbf{D}_N^*$. In Figure 2 we denote the result of this operation as $(\mathbf{g}^*)'$ indicating that this is the complex conjugate of the gain vector before the angular constraints have been applied.

The next block implements the angular constraints. We define an $N\times N$ matrix $\mathbf{S}_{x,y}$ which is zero everywhere except for the intersection of the row corresponding to the constraint x and the column corresponding to the constraint y. The elements of $\mathbf{S}_{x,y}$ can thus be written

$$(\mathbf{S}_{x,y})_{ik} = u'(i-x)u'(k-y), \qquad (10)$$

where $u'(i)$ is 1 if $i=0$ and 0 otherwise. Starting with an $N\times N$ identity matrix, we subtract $\mathbf{S}_{x,x}$ for each constraint x in the set C. A zero is thus placed along the diagonal which, when multiplied with the unconstrained gain vector, will zero the gain in that direction. This yields the constrained gain vector $\mathbf{g}^{*}$. Since we are setting the gain to zero or multiplying by one, the fact that we are manipulating the conjugate of the gain vector does not affect the matching of the constraints. The inverse transform is then applied, yielding an $N\times1$ weight vector $\mathbf{w}'$. This step is performed by pre-multiplication by $\mathbf{D}_N^{-1}$. The resulting unconstrained weight vector matches the angular constraints; however to accomplish this it relies upon the use of antenna signals which do not correspond to available antenna elements.

The final block (imposing the antenna constraints) truncates the $N\times1$ vector $\mathbf{w}'$ to the $K\times1$ weight vector $\mathbf{w}$. This step uses the $K\times N$ matrix $\mathbf{T}_K$, which was defined in the description of the first block for pre-multiplication. Once this step is completed, we have completed the loop and have the new weight vector. Combining all of these blocks, we define the $K\times K$ update matrix $\mathbf{U}$ as the product of all these transformations. The iterative step shown in Figure 2 is then written as

$$\mathbf{w}(l+1) = \mathbf{T}_K\mathbf{D}_N^{-1}(\mathbf{I}-\sum_{x\in C}\mathbf{S}_{x,x})\mathbf{D}_N\mathbf{T}_K^T\mathbf{w}(l) = \mathbf{U}\mathbf{w}(l), \tag{11}$$

where l indicates the iteration index. As l increases, it is desired that the matching of constraints in the angular domain improves until $\mathbf{g}^{*}=(\mathbf{g}^{*})'$ in steady-state. We present an analysis of this algorithm to determine its transient and steady-state properties in the next section.

3. WEIGHT VECTOR ANALYSIS

In this section we analyze the weight vector updating equation to investigate the behavior of the weight vector as the iterative algorithm is applied. The properties used in this derivation are summarized in the Appendix. We may simplify the expression for $\mathbf{U}$ defined by equation 11 as

$$\mathbf{U} = \mathbf{T}_K \mathbf{D}_N^{-1}(\mathbf{I} - \sum_{x \in C} \mathbf{S}_{x,x})\mathbf{D}_N \mathbf{T}_K^T = \mathbf{I} - \sum_{x \in C} \mathbf{T}_K \mathbf{D}_N^{-1} \mathbf{S}_{x,x} \mathbf{D}_N \mathbf{T}_K^T$$

$$= \mathbf{I} - \sum_{x \in C} \mathbf{M}_{x,x}. \tag{12}$$

A new $K \times K$ matrix $\mathbf{M}_{x,x}$ is defined as $\mathbf{T}_K \mathbf{D}_N^{-1} \mathbf{S}_{x,x} \mathbf{D}_N \mathbf{T}_K^T$. The recursive relationship in equation 11 for $\mathbf{w}(l)$ can be replaced with

$$\mathbf{w}(l) = \mathbf{U}^l \mathbf{w}(0). \tag{13}$$

where $\mathbf{w}(0)$ is the initial vector chosen to start the iteration. We will now show that $\mathbf{U}^l$ has the following form:

$$\mathbf{U}^l = \mathbf{I} + \sum_{x \in C} \sum_{y \in C} \mathbf{M}_{x,y} \alpha(l,x,y) = \mathbf{I} + \Sigma\!\!\oplus [\mathbf{M},\ \mathbf{A}(l)]. \tag{14}$$

In equation 14, we have introduced a new matrix operator $\Sigma\oplus$. This operator represents the weighted sum of a set of matrices, in this case $\mathbf{M}_{x,y}$. It and some of its relevant properties, are described in the Appendix. The weighting function $\alpha(l,x,y)$ for the constraints x,y is written in the matrix form $\mathbf{A}(l)$ where the rows correspond to the x constraints and the columns correspond to the y constraints. The major difference between this weighting function and those used in the Appendix is that this function also varies with respect to the iteration index l. To check the correspondence between the form in equation 14 and the value of $\mathbf{U}$ defined in equation 12 we write,

$$\mathbf{U} = \mathbf{I} - \sum_{x \in C} \mathbf{M}_{x,x} = \mathbf{I} - \sum_{x \in C} \sum_{y \in C} \mathbf{M}_{x,y} u'(x-y) = \mathbf{I} + \Sigma\!\!\oplus [\mathbf{M},\ -\mathbf{I}]. \tag{15}$$

Therefore $\mathbf{U}$ follows the form in equation 14 with $\alpha(1,x,y) = -u'(x-y)$ or $\mathbf{A}(1) = -\mathbf{I}$.

Next we examine the inductive step to verify that the product of these matrices is also of the form of equation 14:

$$U^{l+1} = UU^l = (I + \Sigma\!\!\!\!\oplus [M, -I])\,(I + \Sigma\!\!\!\!\oplus [M, A(l)])$$

$$= I + \Sigma\!\!\!\!\oplus [M, A(l)] + \Sigma\!\!\!\!\oplus [M, -I] + \Sigma\!\!\!\!\oplus [M, -I]\,\Sigma\!\!\!\!\oplus [M, A(l)]. \qquad (16)$$

Using the multiplicative property of $\Sigma\oplus$ presented in equation (A.15) to simplify the last term yields

$$U^{l+1} = I + \Sigma\!\!\!\!\oplus [M, A(l)] + \Sigma\!\!\!\!\oplus [M, -I] + \Sigma\!\!\!\!\oplus [M, -IBA(l)]. \qquad (17)$$

The matrix **B** is defined in equation (A.13) of the Appendix. The last three terms in equation 17 can be combined through the use of the linearity property of $\Sigma\oplus$ presented in equation (A.6):

$$U^{l+1} = I + \Sigma\!\!\!\!\oplus [M, A(l) - I - BA(l)] = I + \Sigma\!\!\!\!\oplus [M, A(l+1)]. \qquad (18)$$

This returns U^{l+1} to the form described in equation 14 and verifies that the choice of this form is correct. The last two lines of equation 18 yield a recursive relationship for $A(l)$. This recursion requires the multiplication of two $c_{max} \times c_{max}$ matrices, rather than the multiplication of two $K \times K$ matrices for each iteration required when evaluating U^l directly through successive multiplications by **U**. However, it is also possible to obtain a closed form expression for $A(l)$. From equation 18,

$$A(l+1) = A(l) - I - BA(l) = (I - B)A(l) - I, \qquad (19)$$

with $A(1) = -I$, so that

$$A(l) = -\sum_{i=0}^{l-1} (I-B)^i = -(I-(I-B))^{-1}\,(I-(I-B)^l) = -B^{-1}(I-(I-B)^l). \tag{20}$$

Note that the evaluation of equation 20 requires that **B** be nonsingular. If the eigenvalues of **B** fall between zero and two, A(l) converges to

$$\lim_{l\to\infty} A(l) = -B^{-1}. \tag{21}$$

The steady-state weight vector can thus be written as,

$$\lim_{l\to\infty} w(l) = (I + \sum\!\!\!\!\oplus\, [M,\ -B^{-1}])w(0). \tag{22}$$

4. CONSTRAINT ERROR ANALYSIS

The results in Section 3 demonstrate that the iterative algorithm will converge to a final value given proper conditioning of the matrix **B**. In Section 5 we numerically examine several **B** matrices to confirm the convergence. In this section we investigate how well the solution vector matches the constraints in the angular domain.

The constraint error vector is defined as

$$E(l) = (g^*(l))' - g^*(l) = D_N T_K^T w(l) - (I - \sum_{x \in C} S_{x,x}) D_N T_K^T w(l)$$

$$= \sum_{x \in C} S_{x,x} D_N T_K^T w(l). \tag{23}$$

This vector contains the gains that correspond to each of the constraints and zeros in all other positions. When a correct solution is obtained, this vector should be zero. The two norm of this vector provides a convenient measure of algorithm performance

$$\|E(l)\|^2 = E^H(l)E(l) = (w^H(l)T_K D_N^H \sum_{x \in C} S^H_{x,x})(\sum_{y \in C} S_{y,y} D_N T_K^T w(l))$$

$$= w^H(l)T_K(ND_N^{-1})(\sum_{x \in C}\sum_{y \in C} S_{x,x}S_{y,y})D_N T_K^T w(l). \tag{24}$$

The fact that the product $S_{x,x}S_{y,y}=0$ for $x \neq y$ allows it to be replaced by $S_{x,y}u(x-y)$ and the matrices T_K and D_N^{-1} can also be brought inside the summations yielding

$$\|E(l)\|^2 = Nw^H(l)(\sum_{x \in C}\sum_{y \in C} T_K D_N^{-1} S_{x,y} D_N T_K^T u(x-y))w(l)$$

$$= Nw^H(l)\,\Sigma\oplus[M,\ I]w(l). \tag{25}$$

Substituting equations 13 and 14 for $w(l)$ yields

$$\|E(l)\|^2 = N((I+\Sigma\oplus[M,\ A(l)])w(0))^H\ \Sigma\oplus[M,\ I]\ ((I+\Sigma\oplus[M,\ A(l)])w(0))$$

$$= Nw^H(0)(I+(\Sigma\oplus[M,\ A(l)])^H)\ \Sigma\oplus[M,\ I]\ (I+\Sigma\oplus[M,\ A(l)])w(0). \tag{26}$$

The conjugate transpose of $\Sigma\oplus$M, A(l)] is replaced using the results of equation (A.10) to yield

$$\|E(l)\|^2 = Nw^H(0)(I+\Sigma\oplus[M,\ A^H(l)])\ \Sigma\oplus[M,\ I]\ (I+\Sigma\oplus[M,\ A(l)])w(0). \tag{27}$$

Using the multiplicative property of $\Sigma\oplus$presented in equation (A.15) to simplify the product of the three interior terms yields

$$||\mathbf{E}(l)||^2 = Nw^H(0)\Big(\Sigma\oplus[\mathbf{M},\ \mathbf{I}] + \Sigma\oplus[\mathbf{M},\ \mathbf{IBA}(l)]$$

$$+ \Sigma\oplus[\mathbf{M},\ \mathbf{A}^H(l)\mathbf{BI}] + \Sigma\oplus[\mathbf{M},\ \mathbf{A}^H(l)\mathbf{BBA}(l)]\ \Big)w(0). \tag{28}$$

The four additive terms can be combined through the use of the linearity property of $\Sigma\oplus$ presented in equation (A.6):

$$||\mathbf{E}(l)||^2 = Nw^H(0)\ \Sigma\oplus[\mathbf{M},\ \mathbf{I}+\mathbf{BA}(l)+\mathbf{A}^H(l)\mathbf{B}+\mathbf{A}^H(l)\mathbf{BBA}(l)]\ w(0)$$

$$= Nw^H(0)\ \Sigma\oplus[\mathbf{M},\ (\mathbf{I}+\mathbf{BA}(l))^H(\mathbf{I}+\mathbf{BA}(l))]\ w(0). \tag{29}$$

The transition between the first and second line in equation 29 makes use of the property that $\mathbf{B}=\mathbf{B}^H$ for the matrix $\mathbf{B}$ defined in equation (A.13). The $(\mathbf{I}+\mathbf{BA}(l))$ term is evaluated next. Substitution of $\mathbf{A}(l)$ from equation 20 yields

$$\mathbf{I}+\mathbf{BA}(l) = \mathbf{I}+\mathbf{B}(-\mathbf{B}^{-1}(\mathbf{I}-(\mathbf{I}-\mathbf{B})^l)) = \mathbf{I}-\mathbf{I}+(\mathbf{I}-\mathbf{B})^l = (\mathbf{I}-\mathbf{B})^l. \tag{30}$$

The fact that the matrix specified by equation 30 is Hermitian allows the product $((\mathbf{I}-\mathbf{B})^l)^H(\mathbf{I}-\mathbf{B})^l$ to be written as $(\mathbf{I}-\mathbf{B})^{2l}$. The expression for the error becomes

$$||\mathbf{E}(l)||^2 = Nw^H(0)\,\Sigma\oplus[\mathbf{M},\ (\mathbf{I}-\mathbf{B})^{2l}]w(0). \tag{31}$$

If the eigenvalues of $\mathbf{B}$ lie between zero and two, as required for convergence of equation 21, the $(\mathbf{I}-\mathbf{B})^{2l}$ term approaches zero as l approaches infinity. The steady-state error is therefore,

$$\lim_{l\to\infty} ||\mathbf{E}(l)||^2 = 0 \tag{32}$$

which demonstrates that the algorithm will converge to a solution which meets the specified *constraints.*

5. COMPUTER EVALUATIONS

In this section we present numerical evaluations of the analytical results derived in Sections 3 and 4 using constraints obtained for a sample jammer scenario. The signal example we present consists of an eight element array with $d/\lambda=0.5$ and a 10dB desired signal located at 0°. All powers (in dB) are with respect to the receiver noise power, 0dB. There are six jammers: 20dB at 35°, 20dB at 40°, 20dB at 45°, 20dB at −10°, 30dB at −60°, and 30dB at −70°. The set of angular constraints corresponds to a desired gain of zero in each of the jammer directions, while the gain in the direction of the desired signal is left unconstrained. We only present the results from this test scenario, although several other examples have been tested which yield similar results. These results have been omitted to save space.

We begin by computing the eigenvalues of the matrix **B** defined in equation (A.13) for different DFT lengths N. We use a wide range of N's to demonstrate the algorithm's dependence on the padded signal's length. For the eight element antenna array under consideration, $K=8$. The IMSL[9] routine EIGCH was used to compute the eigenvalues and they are presented to three significant digits in Table 1. It should be noted that with $N=8$ and $N=16$, the resolution in the angular domain is so coarse that two of the constraints fall in the same angular bin. The dimensionality of **B** is thus 5x5 instead of 6x6. Also, for $N=8$, the fact that the eigenvalues are all equal to one indicates that the algorithm converges after one iteration. This can be explained through the fact that when $N=K$, no truncation of the weighted vector is required to meet the antenna constraints. However, it must be realized that although the constraints in the discrete angular domain have be met, the coarse sampling in angle may not provide an adequate null in the desired direction in the continuous domain. As equation (A.13) indicates, the elements of **B** and thus its eigenvalues are roughly proportional to $1/N$. Though this is not an exact relationship, the data presented in Table 1 display this trend. As N is increased to improve the angular resolution, the eigenvalues decrease causing a reduction in the rate at which the error term decays. Also note that all of the eigenvalues listed in Table 1 fall in the range (0,2) guaranteeing convergence.

It is also instructive to examine the transient characteristics of this algorithm. The ratio of the maximum to minimum eigenvalues of the matrix **B** ranged from 1.0E+0 when $N=8$ to 9.9E+2 when $N=512$. Plots are provided showing the variation of the constraint error, the antenna gain in relevant directions, and the entire antenna gain pattern versus number of iterations for several values of N. In all cases, the starting vector w(0) had elements equal to $1/K$ to provide unity gain for the desired signal located at zero degrees. In Figure 3 we plot the constraint error of equation 31 versus the iteration index l for different values of N. The constraint error goes to zero in a monotonic fashion for increasing l for all values of N, thus confirming the convergence of this algorithm. This figure also indicates that larger N values appear to require more number of iterations to achieve the same constraint error. However, it must be remembered that the constraints used here are from the discrete angular gain domain and are not equal to the desired constraints in the continuous domain. Thus the constraint error does not provide a total picture of the nulling capabilities of the resulting weight vectors. To clearly demonstrate this, it is instructive to plot the gain corresponding to the desired location of the null. The gain versus the number of iterations for the jammer located at 35 degrees is provided in Figure 4. The gains at the 6 jammer angles converge to their final values within 100 iterations for $N=16$. This is yet another demonstration that this algorithm converges. Higher values of N result in slower convergence, but they hold the potential for deeper nulls and better satisfaction of the specified constraints. The non-monotonic nature of the gain as a function of iteration index precludes us from guessing what the asymptotic null depths would be, but inspection of Figure 4. indicates that higher N values can lead to greater null depths. The gain in the direction of the desired signal, 0 degrees, is presented in Figure 5 and indicates that only 6 dB attenuation of the desired signal occurs with this algorithm and $N=16$. For larger values of N, the attenuation is even less over the range of iterations shown.

To complete our discussion of this scenario, two plots showing the antenna gain versus angle are presented in Figures 6 and 7. These plots show the directions of the desired constraints using a triangular symbol along the horizontal axis. Figure 6 shows the initial antenna pattern and the patterns obtained after 10 and 10,000 iterations with $N=16$. Figure 7 shows the same information when $N=256$. Comparing these two figures, we note that after 10,000 iterations, $N=256$ results in more accurate null placement then when $N=16$. This is a direct consequence of the finer angular domain resolution achievable with the higher N value. From Figure 7, we see that the initial antenna pattern does not have the nulls at the desired locations. It is clear from the figure that the antenna pattern obtained after 10,000 iterations exhibits deeper nulls at the desired locations compared to the pattern after only 10 iterations.

6. CONCLUSIONS

This paper has described the application of the iterative error reduction algorithm to the problem of calculating a weight vector which meets constraints specifying jammer locations and antenna size. The repetitive nature of the Fourier transforms required to implement this algorithm make it well suited to an optical processing system. A discrete analysis of this technique's transient and steady-state properties has been presented. We have developed criteria for convergence of this algorithm and have numerically demonstrated its convergence. Although the analysis was provided for the discrete case, it is applicable to the continuous optical Fourier transform when the number of antenna elements is large. The fact that 2-D transforms are easily obtained with optics implies that this algorithm may be applied to antennas of arbitrary planar geometry. A Discrete Fourier transform may also be obtained optically through the use of masks and matrix operations to provide additional flexibility in the implementation of this algorithm.

7. ACKNOWLEDGMENT

The authors wish to acknowledge the support of this research by the *Air Force Office of Scientific Research*, contract number AFOSR-84-0239.

8. APPENDIX: DETAILED DERIVATIONS

In this Appendix, we define the matrix operation $\Sigma\oplus$ as well as provide a list of its properties that are used in the derivations of Section 2.

8.1. Definition of the Matrix Operation

The result of this operation is a matrix which is the weighted sum of a two dimensional set of matrices. Symbolically this operation is written as

$$\Sigma\oplus[\mathbf{M},\mathbf{F}] = \sum_{x\in C}\sum_{y\in C} \mathbf{M}_{x,y} f(x,y). \tag{A.1}$$

The set of values over which the function is to be evaluated is denoted by C. For the problem under consideration, this set consists of all of the locations where nulls are to be placed, $C=\{c_1,c_2,\ldots,c_{\max}\}$. These are specified in terms of samples in the angular domain. The term **M** is an arbitrary set of matrices $\mathbf{M}_{x,y}$ which correspond to the constraints $x,y\in C$. The matrix **F** consists of elements specified by the scalar valued function $f(x,y)$ evaluated at the constraints $x,y\in C$ as indicated in Figure 8. The rows of **F** correspond to the individual x constraints, while the columns correspond to the y constraints. Thus **F** is a square matrix whose dimension is equal to the number of constraints in C .

8.2. General Properties of the Matrix Operation

Transposition:

$$(\Sigma\oplus[\mathbf{M},\mathbf{F}])^T = (\sum_{x\in C}\sum_{y\in C} \mathbf{M}_{x,y} f(x,y))^T = \sum_{x\in C}\sum_{y\in C} \mathbf{M}^T_{x,y} f(x,y)$$

$$= \Sigma\oplus[\mathbf{M}^T,\mathbf{F}] \tag{A.2}$$

Conjugation:

$$(\Sigma\oplus[\mathbf{M},\mathbf{F}])^* = (\sum_{x\in C}\sum_{y\in C} \mathbf{M}_{x,y} f(x,y))^* = \sum_{x\in C}\sum_{y\in C} \mathbf{M}^*_{x,y} f^*(x,y)$$

$$= \Sigma\oplus[\mathbf{M}^*,\mathbf{F}^*] \tag{A.3}$$

Conjugate Transpose:

$$\Big(\sum\!\!\!\!\!\oplus\,[\mathbf{M},\ \mathbf{F}]\Big)^{H} = \Big(\Big(\sum\!\!\!\!\!\oplus\,[\mathbf{M},\ \mathbf{F}]\Big)^{*}\Big)^{T} = \sum\!\!\!\!\!\oplus\,[\mathbf{M}^{*T},\ \mathbf{F}^{*}]$$

$$= \sum\!\!\!\!\!\oplus\,[\mathbf{M}^{H},\ \mathbf{F}^{*}] \qquad (A.4)$$

Scalar Multiplication:

$$a\sum\!\!\!\!\!\oplus\,[\mathbf{M},\ \mathbf{F}] = \sum\!\!\!\!\!\oplus\,[a\mathbf{M},\ \mathbf{F}] = \sum\!\!\!\!\!\oplus\,[\mathbf{M},\ a\mathbf{F}] \qquad (A.5)$$

Linearity:

$$a\sum\!\!\!\!\!\oplus\,[\mathbf{M},\ \mathbf{F}] + b\sum\!\!\!\!\!\oplus\,[\mathbf{M},\ \mathbf{G}]$$

$$= a\sum_{x\in C}\sum_{y\in C}\mathbf{M}_{x,y}f(x,y) + b\sum_{x\in C}\sum_{y\in C}\mathbf{M}_{x,y}g(x,y)$$

$$= \sum_{x\in C}\sum_{y\in C}\mathbf{M}_{x,y}(af(x,y) + bg(x,y))$$

$$= \sum\!\!\!\!\!\oplus\,[\mathbf{M},\ (a\mathbf{F} + b\mathbf{G})] \qquad (A.6)$$

8.3. Special Properties for Specific Matrices

Let us now examine the matrices that comprise the set M encountered in Section 2 more specifically. The $K\times K$ matrix $\mathbf{M}_{x,y}$, defined by equation 12, is

$$\mathbf{M}_{x,y} = \mathbf{T}_K\mathbf{D}_N^{-1}\mathbf{S}_{x,y}\mathbf{D}_N\mathbf{T}_K^{T}. \qquad (A.7)$$

It is useful to examine the elements of $\mathbf{M}_{x,y}$ directly through simplification of equation (A.7). Using the definitions of the constituent matrices provided in Section 2, we can show that

$$
\begin{aligned}
(\mathbf{M}_{x,y})_{ho} &= \sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N} (\mathbf{T}_K)_{hi}(\mathbf{D}_N^{-1})_{ik}(\mathbf{S}_{x,y})_{kl}(\mathbf{D}_N)_{lm}(\mathbf{T}_K^T)_{mo} \\
&= \sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N} \acute{u}(h-i)(\mathbf{D}_N^{-1})_{ik}\acute{u}(k-x)\acute{u}(l-y)(\mathbf{D}_N)_{lm}\acute{u}(m-o) \\
&= (\mathbf{D}_N^{-1})_{hx}(\mathbf{D}_N)_{yo} \\
&= (\frac{1}{N}\exp(j\frac{2\pi}{N}(h-1)(x-1)))(\exp(-j\frac{2\pi}{N}(y-1)(o-1))) \\
&= \frac{1}{N}\exp(-j\frac{2\pi}{N}((y-1)(o-1)-(h-1)(x-1))). \qquad (A.8)
\end{aligned}
$$

From equation (A.8), it can be seen that the conjugate transpose of $\mathbf{M}_{x,y}$ is obtained by interchanging x and y:

$$\mathbf{M}_{x,y}^H = \mathbf{M}_{y,x}. \qquad (A.9)$$

The relationship in equation (A.9) for the conjugate transpose of $\mathbf{M}_{x,y}$ can now be used to refine the relationship in equation (A.4) for the conjugate transpose of $\Sigma\oplus$Using the fact that $\mathbf{M}_{x,y}^H = \mathbf{M}_{y,x}$, we can rewrite equation (A.4) as

$$
\begin{aligned}
(\Sigma\oplus[\mathbf{M},\ \mathbf{F}])^H &= \Sigma\oplus[\mathbf{M}^H,\ \mathbf{F}^*] = \sum_{x\in C}\sum_{y\in C} \mathbf{M}_{x,y}^H f^*(x,y) \\
&= \sum_{x\in C}\sum_{y\in C} \mathbf{M}_{y,x} f^*(x,y) = \sum_{x\in C}\sum_{y\in C} \mathbf{M}_{x,y} f^*(y,x) \\
&= \Sigma\oplus[\mathbf{M},\ \mathbf{F}^H]. \qquad (A.10)
\end{aligned}
$$

The last major property to be investigated is a form involving the product of $\Sigma\oplus$ operations:

$$\Sigma\!\!\!\!\oplus[\mathbf{M},\ \mathbf{F}]\ \Sigma\!\!\!\!\oplus[\mathbf{M},\ \mathbf{G}] = \Big(\sum_{x\in C}\sum_{y\in C}\mathbf{M}_{x,y}f(x,y)\Big)\Big(\sum_{p\in C}\sum_{q\in C}\mathbf{M}_{p,q}g(p,q)\Big)$$

$$= \sum_{x\in C}\sum_{y\in C}\sum_{p\in C}\sum_{q\in C}\mathbf{M}_{x,y}\mathbf{M}_{p,q}f(x,y)g(p,q). \qquad (A.11)$$

Let us now concentrate on the matrix product $\mathbf{M}_{x,y}\mathbf{M}_{p,q}$ in equation (A.11).

$$(\mathbf{M}_{x,y}\mathbf{M}_{p,q})_{il} = \sum_{k=1}^{K}(\mathbf{M}_{x,y})_{ik}(\mathbf{M}_{p,q})_{kl}$$

$$= \sum_{k=1}^{K}\Big(\frac{1}{N}\exp\big(-j\frac{2\pi}{N}((y-1)(k-1)-(i-1)(x-1))\big)\Big)$$
$$\times\Big(\frac{1}{N}\exp\big(-j\frac{2\pi}{N}((q-1)(l-1)-(k-1)(p-1))\big)\Big)$$

$$= \frac{1}{N}\exp\big(-j\frac{2\pi}{N}((q-1)(l-1)-(i-1)(x-1))\big)$$
$$\times\frac{1}{N}\sum_{k=1}^{K}\exp\big(-j\frac{2\pi}{N}((y-1)(k-1)-(k-1)(p-1))\big)$$

$$= (\mathbf{M}_{x,q})_{il}\Big(\frac{1}{N}\sum_{k=1}^{K}\exp\big(-j\frac{2\pi}{N}(k-1)(y-p)\big)\Big) = (\mathbf{M}_{x,q})_{il}b(y,p). \qquad (A.12)$$

In equation (A.12), we have expanded $(\mathbf{M}_{x,y})_{ik}$ and $(\mathbf{M}_{p,q})_{kl}$ using equation (A.8), removed common factors from the summation, and defined a function $b(x,y)$ such that

$$b(x,y) = \frac{1}{N}\sum_{k=1}^{K}\exp\big(-j\frac{2\pi}{N}(k-1)(x-y)\big) = \sum_{k=1}^{K}(\mathbf{M}_{y,x})_{kk} = \sum_{k=1}^{K}(\mathbf{M}^{*}_{x,y})_{kk}\,. \qquad (A.13)$$

There is an implied dependence of $b(x,y)$ on K and N which is not explicitly shown since these values remain constant throughout the adaptive cycle. The matrix product in equation

(A.11) can thus be written as

$$\mathbf{M}_{x,y}\mathbf{M}_{p,q} = \mathbf{M}_{x,q}b(y,p). \tag{A.14}$$

Equation (A.14) can now be used to complete equation (A.11) as follows:

$$\Sigma\!\oplus[\mathbf{M},\ \mathbf{F}]\,\Sigma\!\oplus[\mathbf{M},\ \mathbf{G}] = \sum_{x\in C}\sum_{y\in C}\sum_{p\in C}\sum_{q\in C}\mathbf{M}_{x,q}b(y,p)f(x,y)g(p,q)$$

$$= \sum_{x\in C}\sum_{y\in C}\mathbf{M}_{x,y}\sum_{q\in C}\sum_{p\in C}f(x,q)b(q,p)g(p,y) = \Sigma\!\oplus[\mathbf{M},\ \mathbf{FBG}]. \tag{A.15}$$

The transition from line one to line two in equation (A.15) is obtained by interchanging *y* and *q*. As with **F** and **G**, **B** is a matrix whose rows(columns) correspond to the function $b(x,y)$ evaluated at different $x(y)$ constraints in the set C. From the definition of $b(x,y)$ in equation (A.13), **B** is a Hermitian matrix. The important feature of the $\Sigma\oplus$ operator used with the choice of **M** defined in equation (A.7) is that it represents a complex set of operations succinctly. On this set of matrices, all the operations are mapped back to forms where only the weighting function has changed, while the original set of matrices is preserved. This property simplifies the task of solving for the response of the iterative algorithm.

9. REFERENCES

1. W.F.Gabriel, "A High-Resolution Target-Tracking Concept Using Spectral Estimation Techniques", Tech. report 8797, Naval Research Lab, May 1984.

2. R.W.Gerchberg and W.O.Saxton, "A Practical Algorithm for the Determination of Phase From Image and Diffraction Plane Pictures", *Optik*, Vol. 35, No. 2, April 1972, pp. 237-246.

3. J.R.Fienup, "Phase Retrieval Algorithms: a Comparison", *Applied Optics*, Vol. 21, No. 15, 1 August 1982, pp. 2758-2769.

4. A.V.Oppenheim, M.H.Hayes, and J.S.Lim, "Iterative Procedures for Signal Reconstruction from Fourier Transform Phase", *Optical Engineering*, Vol. 21, No. 1, January/February 1982, pp. 122-127.

5. D.K.Smith and R.J.Marks II, "Closed Form Bandlimited Image Extrapolation", *Applied Optics*, Vol. 20, No. 14, 15 July 1981, pp. 2476-2483.

6. M.S.Sabri and W.Steenart, "An Approach to Band-Limited Signal Extrapolation: The Extrapolation Matrix", *IEEE Trans. on Circuits and Systems*, Vol. CAS-25, No. 2, February 1978, pp. 74-78.

7. G.Schils and D.W.Sweeney, "Iterative Techniques for the Synthesis of Optical-Correlation Filters", *J. of the Optical Soc. of America - A*, Vol. 3, No. 9, September 1988. pp. 1433-1422.

8. R.J.Marks II, "Coherent Optical Extrapolation of 2-D Band-Limited Signals: Processor Theory", *Applied Optics*, Vol. 19, No. 10, 15 May 1980, pp. 1670-1672.

9. *IMSL Library Reference Manual 9th Ed.*, 7500 Bellaire Blvd., Houston TX, 1982.

LIST OF TABLES

LIST OF FIGURES

N	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
8	1.00E+0	1.00E+0	1.00E+0	1.00E+0	1.00E+0	
16	9.42E-1	7.08E-1	5.00E-1	2.92E-1	5.79E-2	
32	6.71E-1	4.78E-1	2.51E-1	7.75E-2	2.17E-2	6.11E-4
64	3.21E-1	2.34E-1	1.25E-1	5.87E-2	1.12E-2	5.20E-4
128	1.69E-1	1.15E-1	6.29E-2	2.00E-2	7.97E-3	1.44E-4
256	8.19E-2	5.83E-2	3.16E-2	1.21E-2	3.34E-3	1.09E-4
512	4.17E-2	2.88E-2	1.58E-2	5.59E-3	1.80E-3	4.21E-5
1024	2.08E-2	1.45E-2	7.88E-3	2.78E-3	9.09E-4	2.17E-5

Table 1: Eigenvalues of **B** for the APAR Case Study

FIG. 1

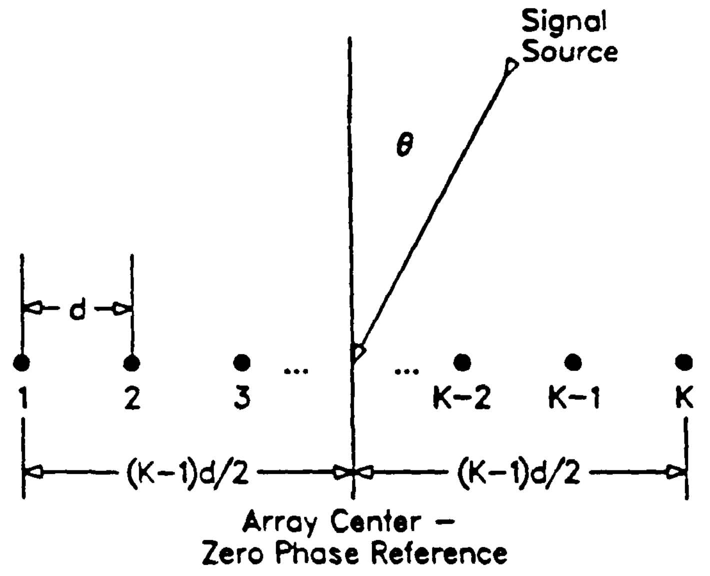
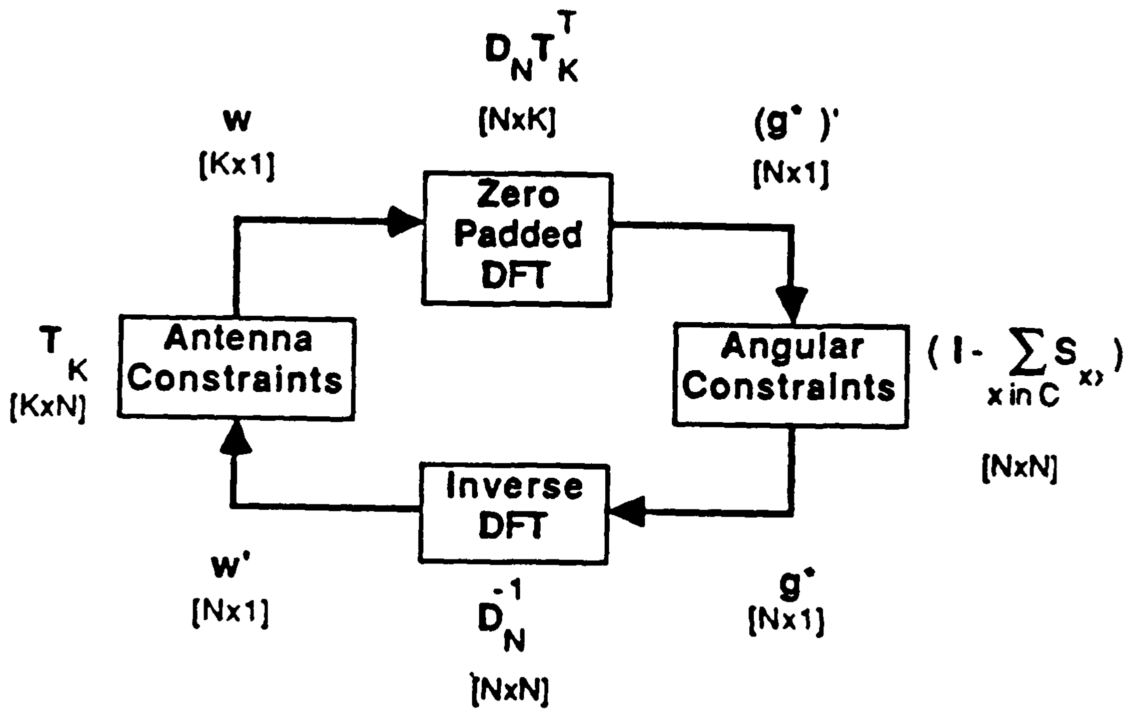
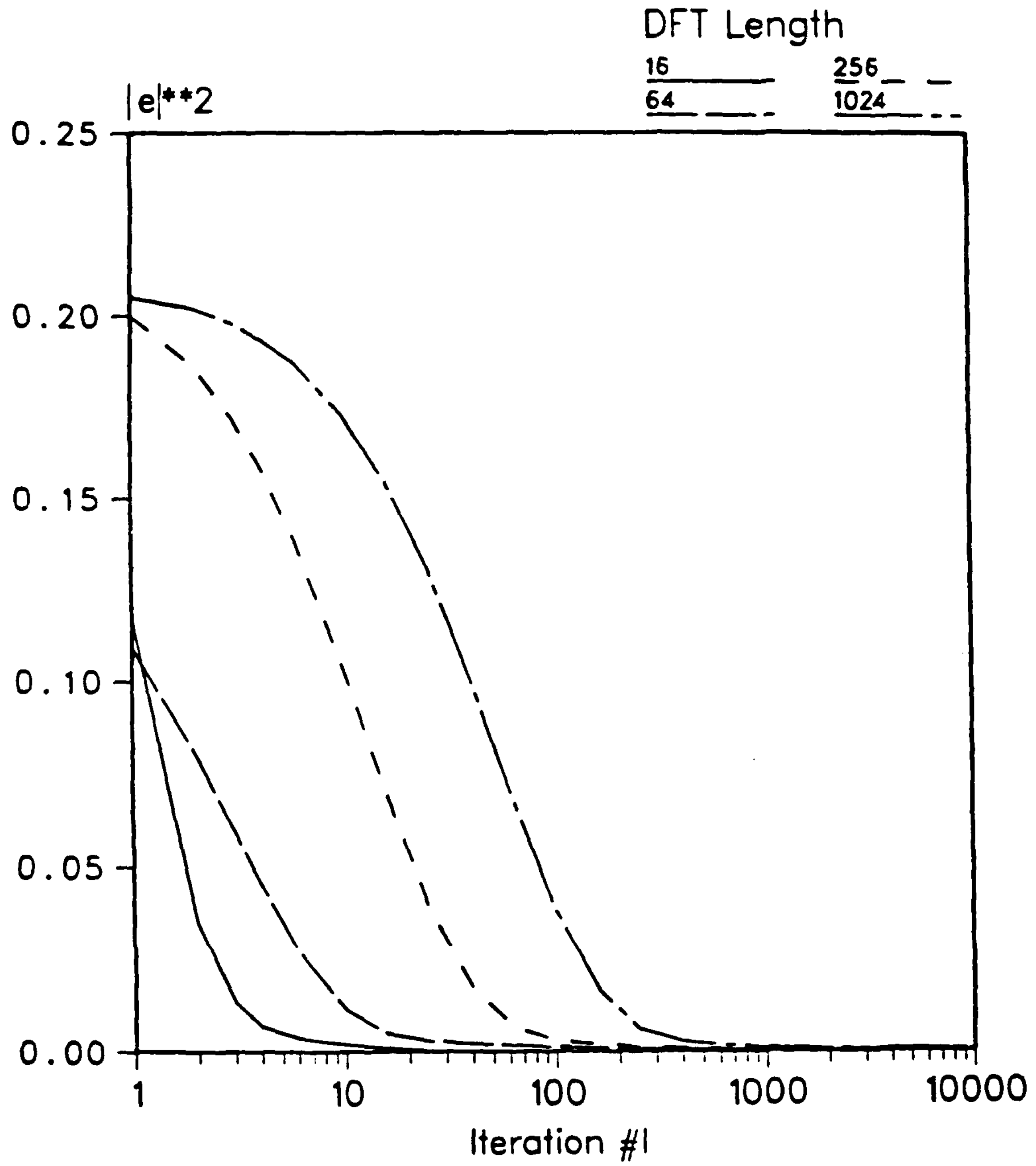

Signal Source
θ
d
1
2
3
...
...
K-2
K-1
K
(K-1)d/2
(K-1)d/2
Array Center - Zero Phase Reference

FIG. 2



$D_N T_K^T$
[NxK]
w
[Kx1]
$(g^*)'$
[Nx1]
Zero Padded DFT
T_K
[KxN]
Antenna Constraints
Angular Constraints
$(I - \sum_{x \in C} S_{x})$
[NxN]
Inverse DFT
w'
[Nx1]
D_N^{-1}
[NxN]
g^*
[Nx1]

FIG. 3



DFT Length
16
64
256
1024
|e|**2
0.25
0.20
0.15
0.10
0.05
0.00
1
10
100
1000
10000
Iteration #I

FIG. 4

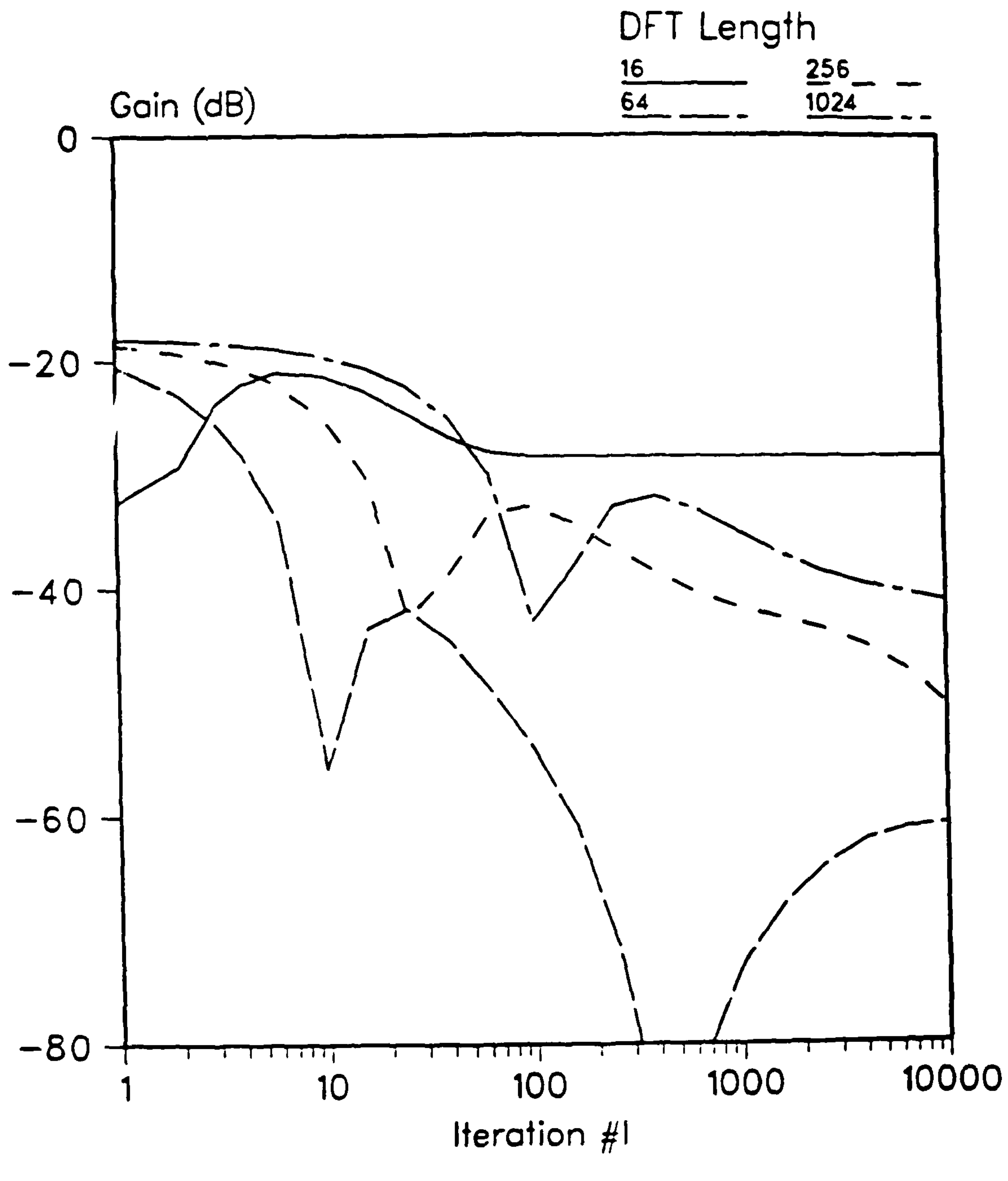
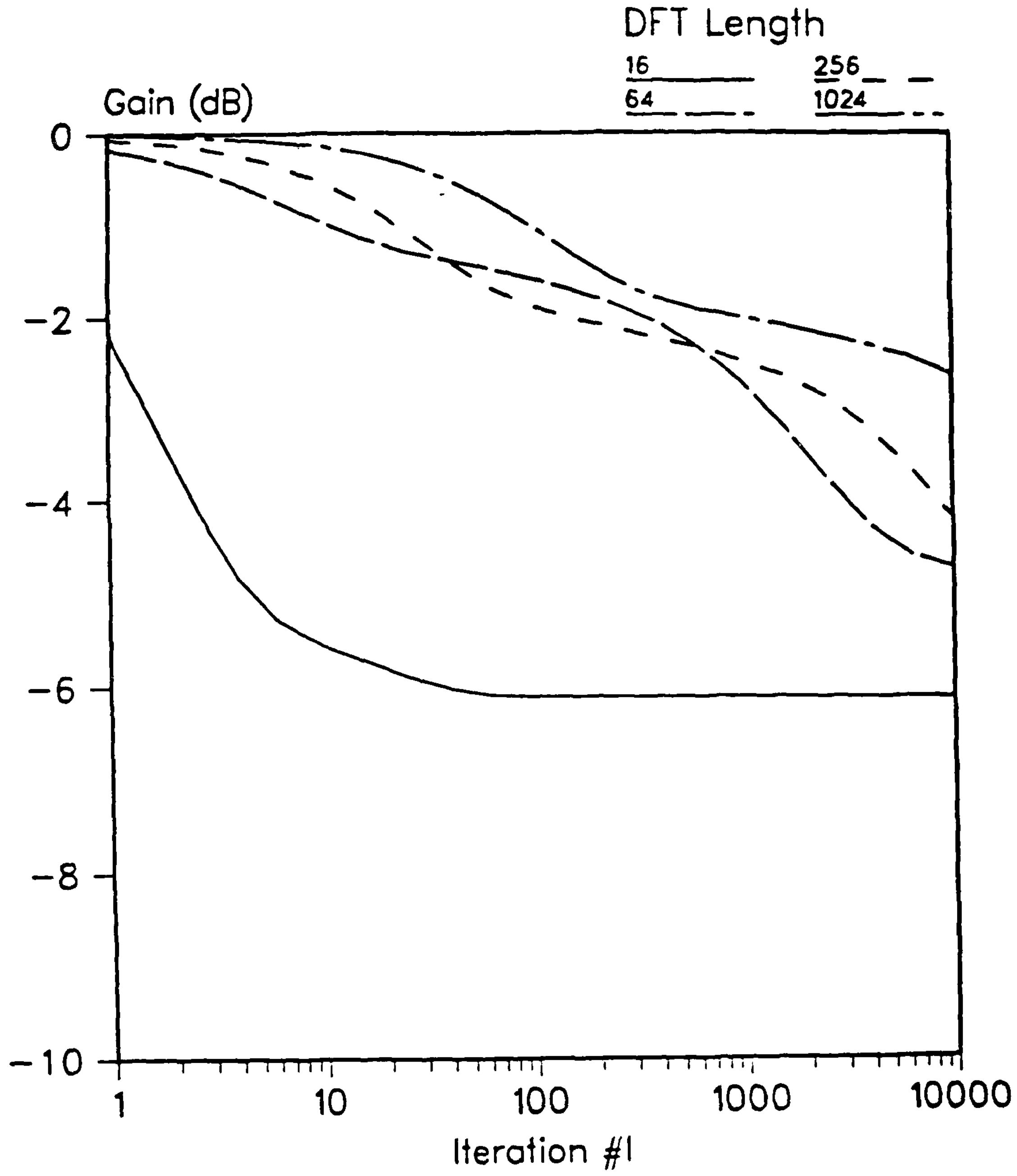
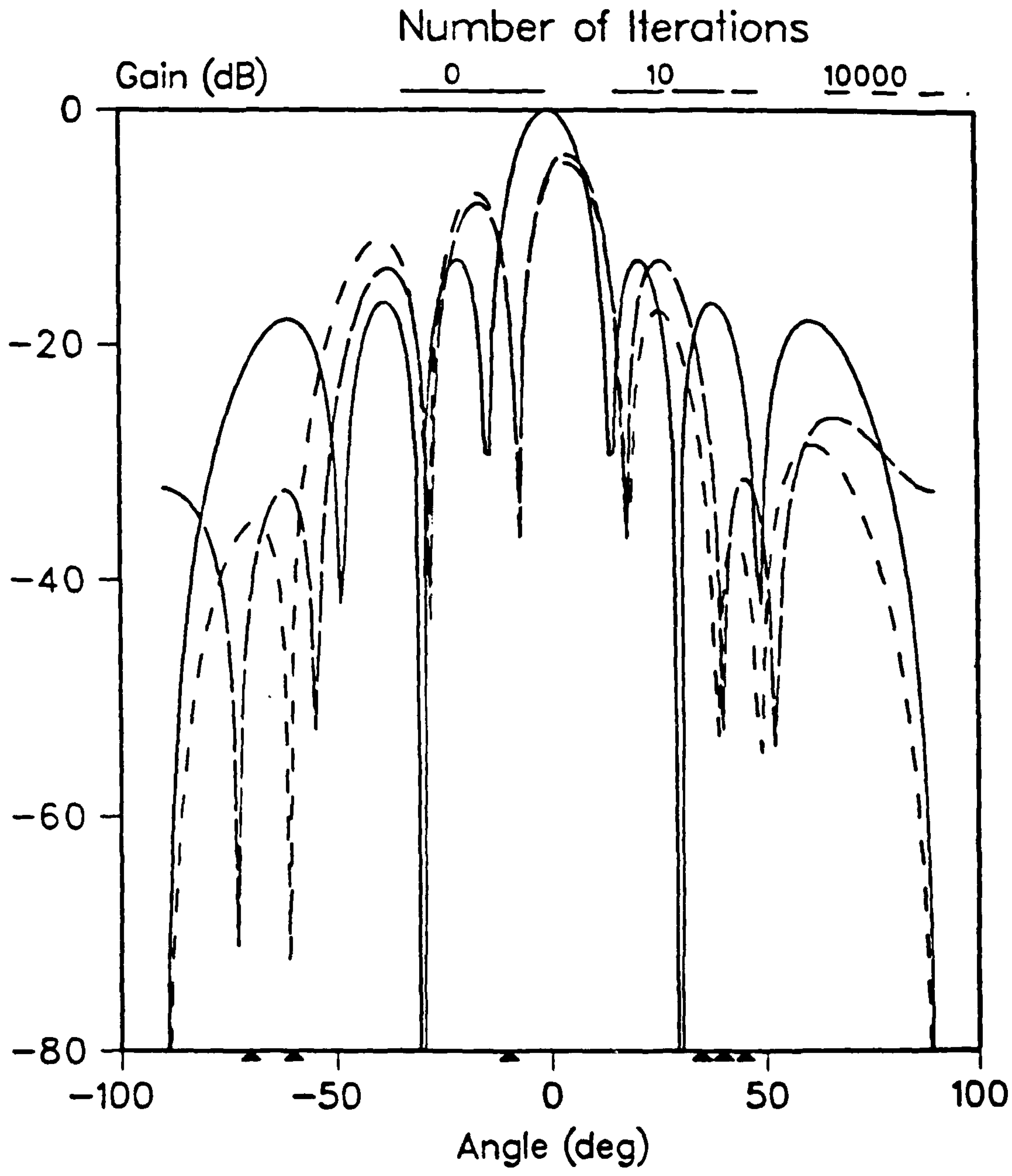

DFT Length
16
256
64
1024
Gain (dB)
0
-20
-40
-60
-80
1
10
100
1000
10000
Iteration #I

FIG. 5



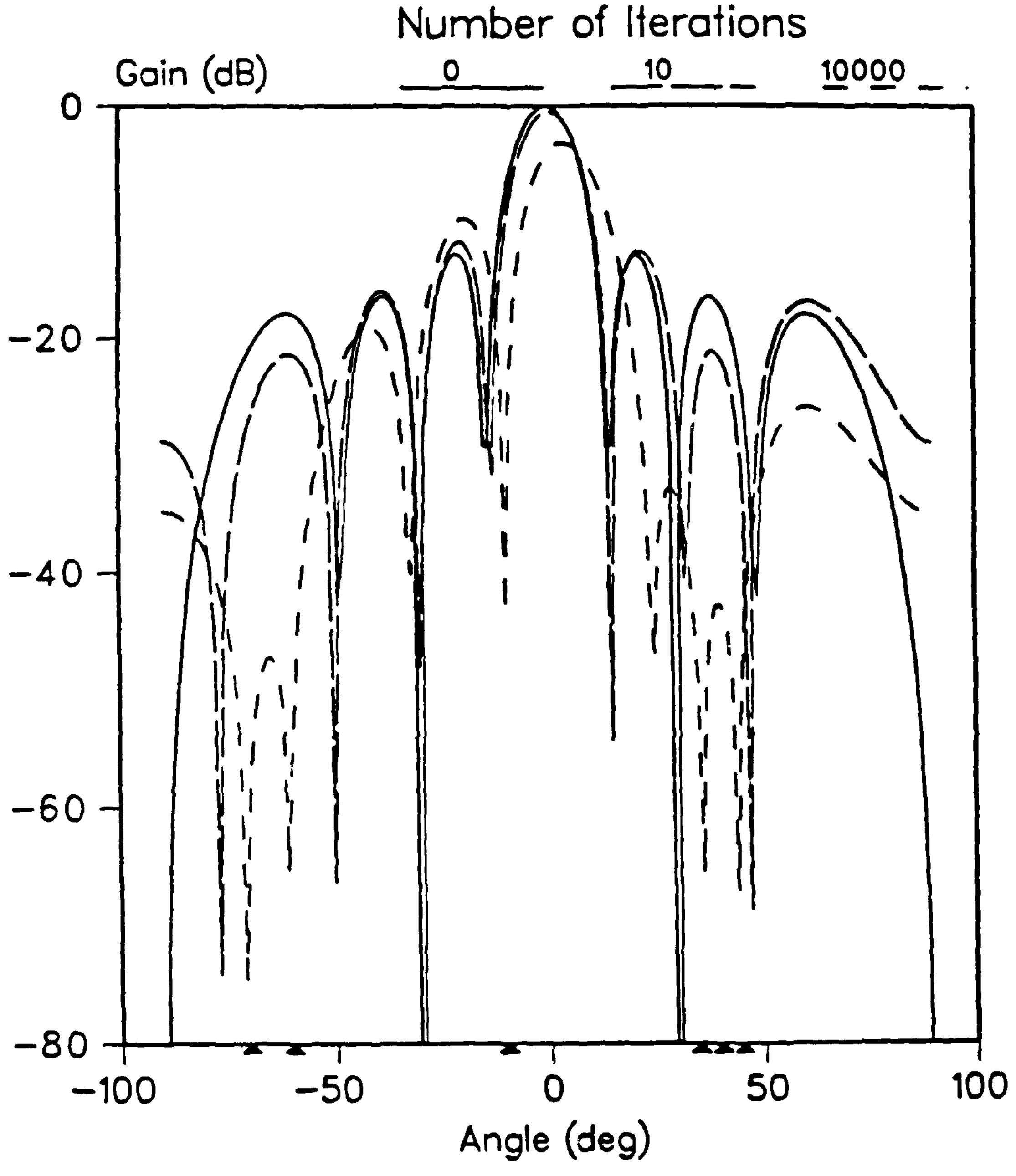
DFT Length
16 ——
256 — —
64 — - —
1024 — - -
Gain (dB)
0
-2
-4
-6
-8
-10
1
10
100
1000
10000
Iteration #l

FIG. 6



Number of Iterations
Gain (dB)
0
10
10000
0
-20
-40
-60
-80
-100
-50
0
50
100
Angle (deg)

FIG. 7



Number of Iterations
Gain (dB)
0
10
10000
0
-20
-40
-60
-80
-100
-50
0
50
100
Angle (deg)

CHAPTER 11:

RECURSIVE MATRIX INVERSE UPDATE ON AN OPTICAL PROCESSOR

Proc. SPIE
Vol. 975
August 1988

Recursive Matrix Inverse Update on an Optical Processor

David P. Casasent and Edward J. Baranoski
Center of Excellence in Optical Data Processing
Electrical and Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A high accuracy optical linear algebraic processor (OLAP) using the digital multiplication by analog convolution (DMAC) algorithm is described for use in an efficient matrix inverse update algorithm with speed and accuracy advantages. The solution of the parameters in the algorithm are addressed and the advantages of optical over digital linear algebraic processors are advanced.

1 Introduction

The optical processor of Figure 1 is considered. It is well detailed [1]. It operates on N-digit encoded data. The digit representation for each of M numbers (a vector) is fed time-sequentially (one digit each T_1) to the M point modulators at P_1. These point modulators are imaged onto M regions of a multi-channel (AO) cell at P_2. This cell is fed with the N digits of a second vector with one number in this vector entered each $T_2 = NT_1$ to the N channels at P_2. The light leaving P_2 is the product of the P_1 data and the contents of P_2. This is integrated vertically onto N detectors at P_3. The contents of P_3 are shifted by one digit each T_1 and the new data incident on P_3 is added to the prior shifted P_3 data. Each of the M channels of the system produces the convolution of the data fed to one P_1 point modulator and the contents of the M regions of P_2. By the DMAC algorithm [2-4], this is the mixed-radix product of the two numbers. It is easily converted to a binary representation by an output A/D, shift register, and adder. In our system, separate A/Ds are present on each detector (to reduce A/D dynamic range requirements). The full system performs M multiplies and additions every T_2; i.e. an M-element vector inner product on N-digit words each T_2. The system can handle over N-digit accuracy by bit-partitioning [5,6] using base $B > 2$ data [1]. It can handle vectors with dimensions larger than M by diagonal partitioning [1]. It handles bipolar numbers by a negative base representation [7].

Section 2 reviews the matrix inverse update algorithm we use. Section 3 presents data on the use of this algorithm on the processor of Figure 1 for adaptive phased array radar (APAR). Section 4 compares this optical system to digital processors.

2 APAR Covariance Update Algorithm

In APAR, the weights $\mathbf{w}$ for a scenario described by a covariance matrix $\mathbf{R}$ and a steering vector $\mathbf{s}$ are given by

$$\mathbf{w}(k+1) = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k+1)\mathbf{s}^* \tag{1}$$

where $(k+1)$ denotes the present output time sample used. We update the $\hat{\mathbf{R}}_{\mathbf{xx}}$ estimate as new data enters

$$\hat{\mathbf{R}}_{\mathbf{xx}}(k+1) = (1-\beta)\hat{\mathbf{R}}_{\mathbf{xx}}(k) + \beta \mathbf{x}^{*}\mathbf{x}^{T} \tag{2}$$

where $0 \leq \beta < 1$ is a weighting factor adjusted to give more weight (larger β) to new data (this is used when the environment changes rapidly) or more weight to prior data (if desired). Calculation of $\hat{\mathbf{R}}_{\mathbf{xx}}$ is time consuming, as is calculation of its inverse (this is required for every time sample vector $\mathbf{x}$ received at the array). Thus we consider updating the prior $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}$ inverse directly using [8]

$$\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k+1) = \frac{1}{(1-\beta)}\left\{\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k) - \frac{\left[\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k)\mathbf{x}^{*}(k+1)\right]\left[\mathbf{x}^{T}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k)\right]}{(1-\beta)/\beta + \mathbf{x}^{T}(k+1)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k)\mathbf{x}^{*}(k+1)}\right\}. \tag{3}$$

For each new $\mathbf{x}$ set of data received at the array elements at time $(k+1)$ we form the vector $\mathbf{p}(k+1) = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k)\mathbf{x}^{*}(k+1)$. Then (3) is easily calculated as

$$\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k+1) = \frac{1}{(1-\beta)}\left\{\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(k) - \frac{\mathbf{p}(k+1)\mathbf{p}^{H}(k+1)}{(1-\beta)/\beta + \mathbf{x}^{T}(k+1)\mathbf{p}(k+1)}\right\}. \tag{4}$$

This algorithm is attractive since it is $O[N^2]$ vs. $O[N^3]$ for calculating the matrix inverse. This algorithm does not require calculation of $\mathbf{R}$ nor direct calculation of its inverse. It also requires fewer bits of accuracy than calculation of $\mathbf{R}^{-1}$ from $\mathbf{R}$.

3 Data Tests

We tested this algorithm and our processor for an $N = 8$ element linear antenna array with spacings $d = 2\lambda$ between elements, a mainlobe width of $\pm 3°$ and grating lobes at $\pm 30°$. For all cases the signal power is our 0dB reference point, the signal is at $+10°$, and the antenna noise is -20dB (this is a typical value for actual cases). The jammers (directional interference) vary in number and strength (Table 1). The performance measure(s) we consider are SNIR (this is the best measure) vs. time sample k and null depth (at $k = 40$). The condition number C varies up to 10^7 for the different scenarios (λ_{max} values correspond to jammers and are larger for stronger jammers, and λ_{min} and small values correspond to antenna noise). Recall that $\mathbf{R}$ is normalized by $N/\mathrm{Tr}[\mathbf{R}]$ to keep the maximum element one and $\sum \lambda_n = N$. This reduces λ_{min} and increases C when new jammers are present. With many jammers at the same dB level, all have the same lower λ_{max} value but λ_{min} is reduced more and thus C increases.

We first consider the accuracy required. We use case 5 and plot SNIR vs. k with $\beta = 0.2$ and $\hat{\mathbf{R}}_{\mathbf{xx}}(0) = 1000\mathbf{I}$. The results for a 16-bit system (Figure 2a) show unstable oscillations. The 20- and 32-bit systems (Figures 2b and 2c) show the same SNIR essentially equal to that obtained with standard double precision $\mathbf{R}^{-1}$ algorithms (Figure 2d). Thus our matrix inverse update algorithm requires only a 20-bit processor (compared to 24-bits required with standard $\mathbf{R}^{-1}$ algorithms). This is significant, since the calculations on our optical system are then faster (since fewer bits are required) and the optical system is cheaper (since fewer AO channels, detectors, and A/Ds are required). Figure 3 shows the power, SNIR, and antenna plots obtained with a standard UDU^H $\mathbf{R}^{-1}$ algorithm (Figures 3a, 3c, and 3e) and our matrix inverse update algorithm (Figures 3b, 3d, and 3f) with $\beta = 0.2$ and $B = 20$ bit accuracy for case 5 in Table 1.

The speed of convergence of our algorithm can be controlled by selecting β. Smaller $\beta = 0.05$ values weight new data samples less and larger $\beta = 0.2$ values weight new data samples more. Thus small (large)

Table 1: Jammer Scenarios

	Jammer Strengths (dB)					Cond.	
Cases	$\theta = -10°$	$-5°$	$0°$	$0.5°$	$9°$	No.	Remarks
1	-10					800	
2	0	-10				924	1-5 Jammers
3	0	0	-10			1045	Max.=0dB
4	0	0	0	-10		1133	
5	0					921	
6	0	0				1044	1-5 Jammers
7	0	0	0			1110	Max.=0dB
8	0	0	0	0		1677	
9	10					8020	
10	0	10				8050	1-5 Jammers
11	0	0	10			8060	Max.=10dB
12	0	0	0	10		8814	
13	0	0	0		0	1555	Mainbeam Jammer
14	0	0	0	40		8×10^6	Strong jammer

β provides better (poorer) estimate of **R** and smoother (more erratic) changes with k. We expect faster convergence with larger β but better SNIR and an antenna pattern near the optimum with smaller β. We also thus expect small β to allow use of fewer bits (B). Figures 4-8 show SNIR and the antenna pattern for different cases and β values. These also show that our algorithm performs well for various scenarios: multiple jammers (Figure 4), jammers spaced only 0.5° apart (Figure 5), multiple jammers providing a large condition number $C = 8800$ (Figure 6), multiple jammers with a mainbeam jammer (Figure 7), and one very strong jammer out of four with a very large condition number (Figure 8). Figure 9 illustrates the effect of a single blinking jammer at $-10°$. The jammer starts in the off state and remains off for the first $k = 20$ samples. The jammer state then changes every 20 samples (in Figure 9, the jammer is on during $k = 20 \rightarrow 40$ and during $k = 60 \rightarrow 80$). This example illustrates the importance of using a β value to reduce the effects of prior data samples. Uniform weighting is achieved by setting $\beta = 1/k$, whereas exponential weighting requires a constant value for β thus reducing the effects of prior data with respect to the current data sample. Figures 9a and 9c show the SNIR and antenna pattern for uniform weighting, and Figures 9b and 9d show the same plots for exponential weighting. Exponential weighting is seen to respond more quickly to the blinking jammer.

4 Optical vs. Digital Linear Algebraic Processors

DMAC architectures have been stated [9] to be unattractive on the basis of the number of multiplies per A/D conversion they perform. This ratio is typically less than one. We show that it can exceed one for our processor and that this is not a valid measure. We then advance a preferable measure and show that DMAC architectures improve faster than digital processors as equivalent technology advances occur and that at present, the DMAC architecture exceeds the throughput of digital multipliers and adders.

Table 2: System performance for various cases using 50MHz A/Ds and a 16-channel AO cell to perform 16-bit multiplications

Base L	Digits N_L	A/D bits N_d	M	T_2 (nsec)	MOPS	$\frac{\text{Operations}}{\text{A/D conv.}}$
2	16	5	31	320	97	0.12
4	8	9	56	160	350	0.88
		10	62		388	0.97
8	6	9	10	120	83	0.28
		12	83		692	2.31
16	4	9	2	80	25	0.13
		12	18		225	1.13
		15	125		1563	7.81

We first analyze the performance and requirements of the optical system. The system of Figure 1 performs M multiplies and M additions each T_2, or M/T_2 operations per second where an operation is a multiply and an addition. The number of A/D bits required for a base L system is $\log_2[M(L-1)^2]$. Tables 2 and 3 list the performance of this system for 50 and 100MHz A/Ds and a 16-channel AO cell for different values of the base L, different numbers of point modulators M, and with the number N_A of AO channels required equal to the N_L value given. All parameter values given are possible with present technology. These tables show that performance above 3 GOPS is possible and that the number of operations per A/D conversion can exceed one (thus negating earlier arguments [9]). We assumed T_1 =20 and 10nsec, respectively, and an AO cell with aperture time T_A=10μsec. The number of processing channels M is bounded by the aperture time as

$$M \leq T_A/N_L T_1 \tag{5}$$

and by the bit resolution N_d of the A/D as

$$M \leq 2^{N_d}/(L-1)^2. \tag{6}$$

The largest value for M is desired for the maximum number of operations per second (OPS). This occurs when

$$\frac{T_A}{N_L^2 T_1^2} = \frac{2^{N_d}}{N_L T_1 (2^{n/N_L}-1)^2} \text{or} \tag{7}$$

$$\frac{1}{N_L}(2^{n/N_L}-1)^2 = \frac{T_1}{T_A}2^{N_d}. \tag{8}$$

We maximize the number of OPS by selecting $N_L \geq \log_L 2^n$ using (7).

The speed of both digital and optical systems can be increased through parallelism. For example the multiplication of two 16-bit numbers can be achieved digitally using arrangements of 8-bit multipliers and adders. If one 8-bit multiplier operates in time T, then various combinations of these multipliers and adders can perform 16-bit multiplication in times ranging from T to $4T$. However the speed increases *linearly* with the number of components and the *operations per second (OPS) per component is a constant* ($1/4T$ for the above example) regardless of the architecture. Bit partitioning in DMAC also allows any desired accuracy

Table 3: System performance for various cases using 100MHz A/Ds and a 16-channel AO cell to perform 16-bit muliplications

Base L	Digits N_L	A/D bits N_d	M	T_2 (nsec)	MOPS	Operations / A/D conv.
2	16	5	62	160	388	0.24
4	8	9	56	80	700	0.88
		11	125		1563	1.95
8	6	9	10	60	167	0.28
		12	83		1383	2.31
		13	166		2787	4.61
16	4	9	2	40	50	0.13
		10	4		100	0.25
		12	18		450	1.13
		15	145		3625	9.06

$N_L > N_A$ to be achieved. As Tables 2 and 3 showed, one can optically achieve high accuracy with low accuracy components (e.g., 16-bit multiplies with 5-bit A/Ds). Optically, there is a much better OPS per component ratio than occurs digitally. Specifically, to digitally form xn-bit products using x-bit devices, the OPS per component ratio is $1/x^2$ and thus decreases rapidly as the accuracy required increases. For example, if two digital systems each capable of a certain number of OPS were paralleled, the number of OPS doubles but the number of OPS per component remains the same. Since a system's speed is the OPS per component times the number of components, we now consider the OPS per component of digital and optical systems, where a digital component is a multiplier and an optical component is an A/D (with the same resolution assumed for both the multiplier and the A/D). This is a better measure than operations per A/D conversion [9]. Since the technologies for parallel high-speed digital multipliers and A/D converters are similar, assuming the same resolution for each is realistic and allows us to project performance for improved device speeds (for both digital multipliers and A/D converters).

We now derive an expression for the OPS per component of the optical DMAC architecture of Figure 1. DMAC improves faster than the linear increase with component speed that occurs digitally. To see this recall that Figure 1 achieves M/T_2 OPS. Since $T_2 = N_L T_1$, the term $1/T_2$ increases linearly with device speed. However, M also increases with device speed as in (5). Optimum performance occurs when M is maximized, which occurs at the crossover point defined by (7). Figure 10 shows the maximum allowable M from (5) for different A/D speeds (the ascending curves) and for the A/D resolution N_d constraint on M from (6) for different A/D resolutions (the descending curves). The intersections of these curves defines the maximum allowable M, which is seen to increase as device speed or resolution improves. The base L (shown along the horizontal axis) is determined by the crossover point (Figure 10 was produced for 16-bit multiplications). Figure 11 shows the maximum M (when L is chosen as in Figure 10) as a function of A/D speed for 9, 12, and 15-bit A/Ds. Since the system's overall OPS is $M/N_L T_1$, we see that M increases with component speed (Figure 11) and that $1/T_1$ increases linearly with component speed. Thus, the OPS for this optical architecture increases nearly as the *square* of the component (A/D) speed, while the OPS in a digital system increase only *linearly* with component (multiplier) speed.

To compare optical and digital performance, we consider the ratio OPS/component of the optical system divided by the OPS/component of the digital system. Figure 12 shows this data for 12, 24, and 36-bit multiplications as a function of component speed for 12-bit devices (digital multipliers and A/D converters). As seen, the optical system is preferable for component speeds above 1.5MHz (easily available now) and the advantage of the optical system over the digital system becomes *better* as component speed increases.

5 Acknowledgment

We gratefully acknowledge the support of General Electric (Contract F14-M31350) and AFOSR for the research reported upon in this paper.

References

[1] D. Casasent and B. K. Taylor, "Banded-matrix high-performance algorithm and architecture," *Applied Optics*, vol. 24, pp. 1476–80, May 15 1985.

[2] E. Swartzlander, "The quasi-serial multiplier," *IEEE Trans. Comp.*, vol. C-22, pp. 317–321, April 1973.

[3] H. Whitehouse and J. Speiser, "Linear signal processing architectures," in G. Tacconi, ed., (*Aspects of Signal Processing - Part II*), (Boston, MA), NATO Advanced Study Institute, 1976, pp. 669–702.

[4] D. Psaltis, D. Casasent, D. Neft, and M. Carlotto, "Accurate numerical computation by optical convolution," *Proc. SPIE*, vol. 232, pp. 151–56, April 1980.

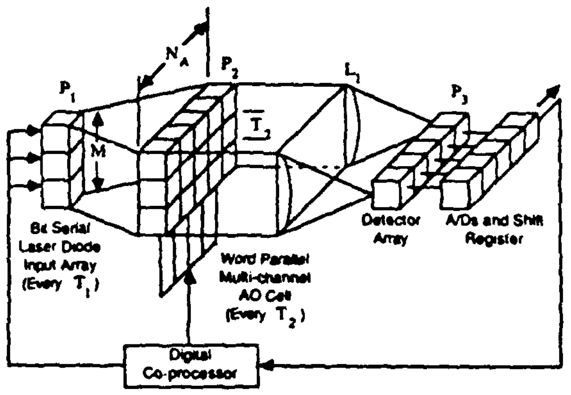
[5] D. Casasent and S. Riedl, "Time and space integrating optical laboratory matrix-vector array processor," *Proc. SPIE*, vol. 698, August 1986.

[6] D. Casasent and S. Riedl, "Direct finite element solution on an optical laboratory matrix-vector processor," *Optical Communications*, vol. 65, pp. 329–333, 1 March 1988.

[7] C. Perlee and D. Casasent, "Negative-base encoding in optical linear algebra processors," *Applied Optics*, vol. 25, pp. 168–69, January 15 1986.

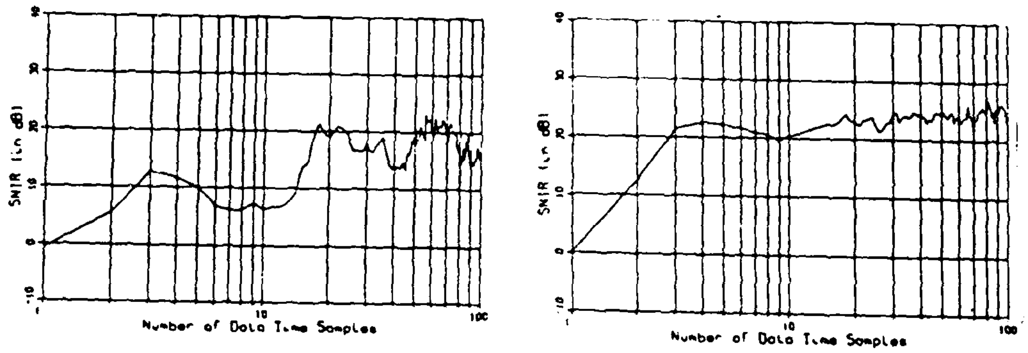
[8] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. New York: J. Wiley & Sons, Inc., 1980, pp. 324-326.

[9] D. Psaltis and R. A. Athale, "High accuracy computation with linear analog optical systems: a critical study," *Applied Optics*, vol. 25, pp. 3071–3077, September 15 1986.



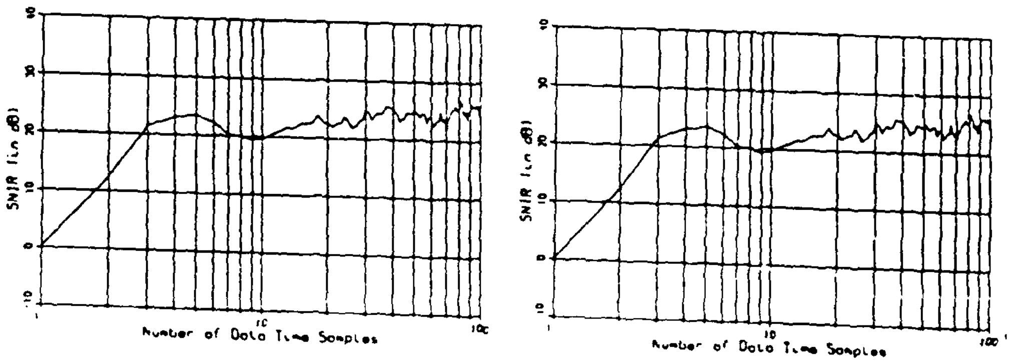
N_A
P_1
P_2
L_1
P_3
T_2
M
Bit Serial Laser Diode Input Array (Every T_1)
Word Parallel Multi-channel AO Cell (Every T_2)
Detector Array
A/Ds and Shift Register
Digital Co-processor

Figure 1: Multi-channel high accuracy time and space integrating architecture



(a) 16 bits (unstable)

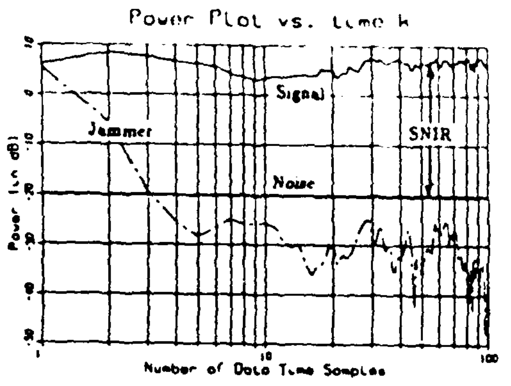
(b) 20 bits (stable)



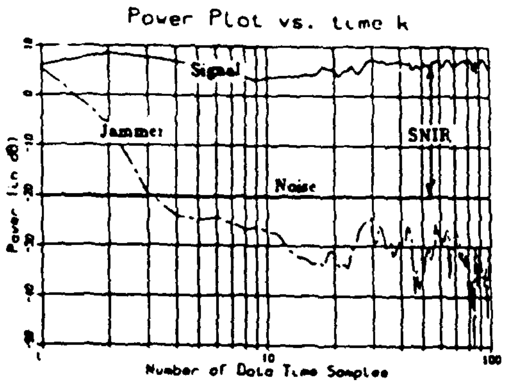
(c) 32 bits (stable, same as (b))

(d) Double precision standard $\mathbf{R}^{-1}$ algorithm

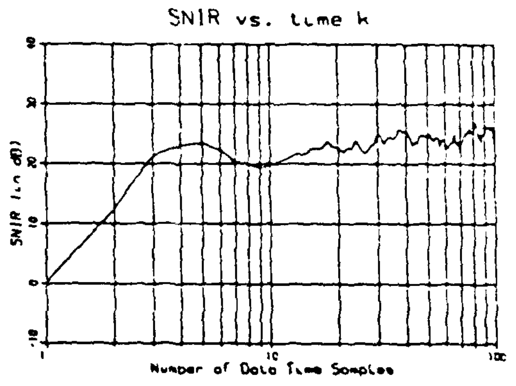
Figure 2: Accuracy required (case 5)



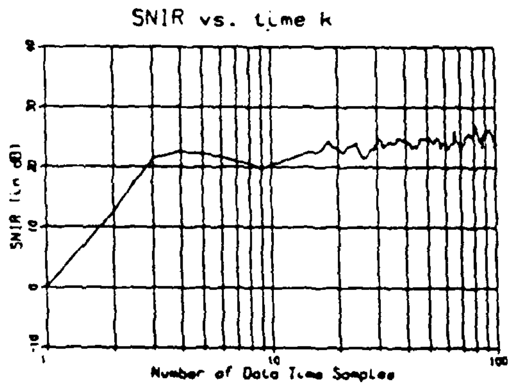
(a) Power



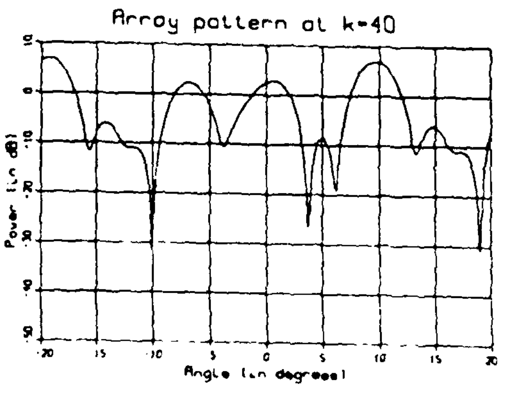
(b) Power



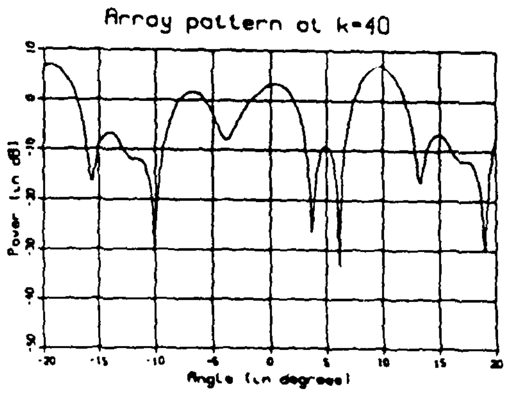
(c) SNIR



(d) SNIR

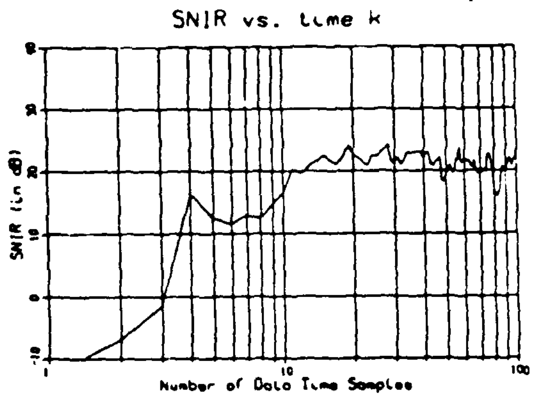


(e) Antenna output

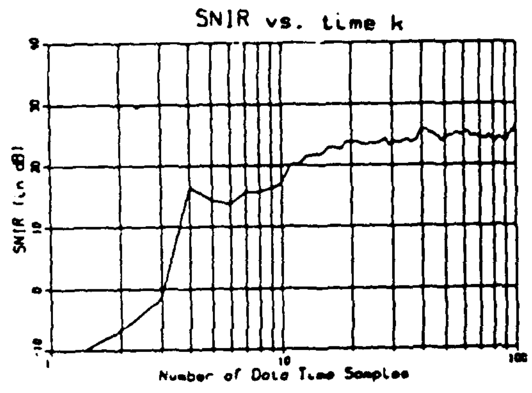


(f) Antenna output

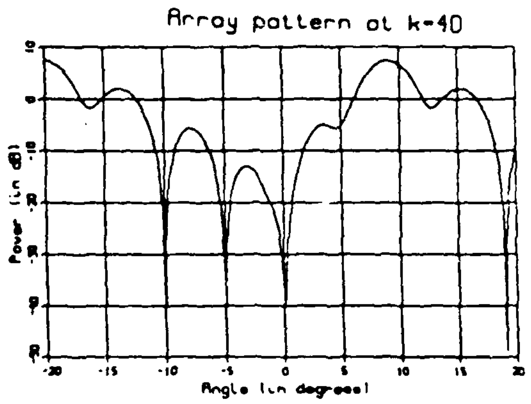
Figure 3: Same results obtained for our algorithm (b, d, and f) and UDU^H (a, c, and e)



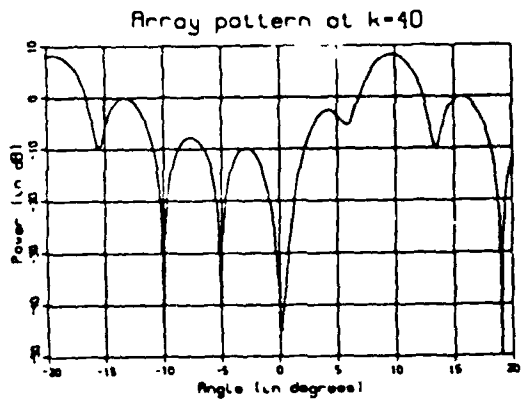
(a) SNIR for $\beta = 0.2$



(b) SNIR for $\beta = 0.05$

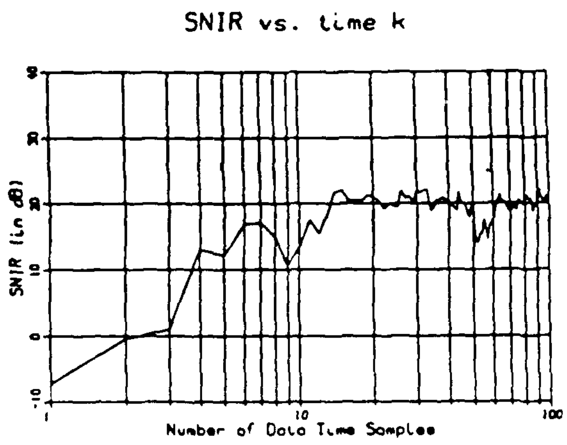


(c) Ant. pattern for $\beta = 0.05$

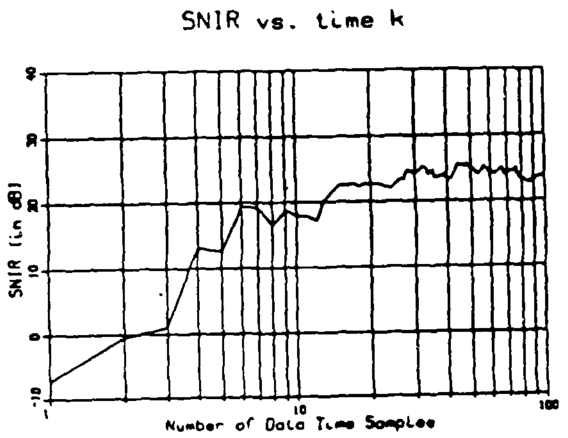


(d) Ant. pattern for $\beta = 0.05$

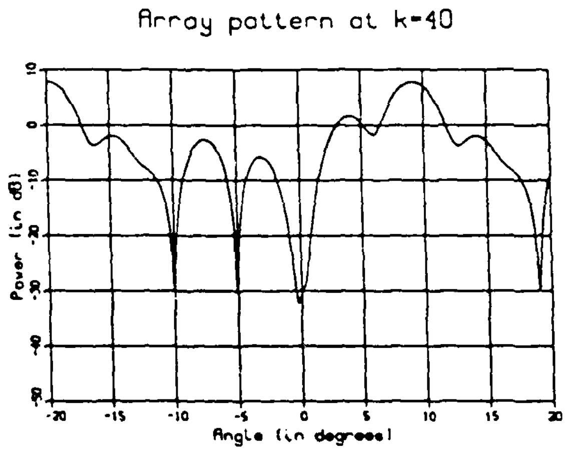
Figure 4: Multiple jammer scenario (case 4)



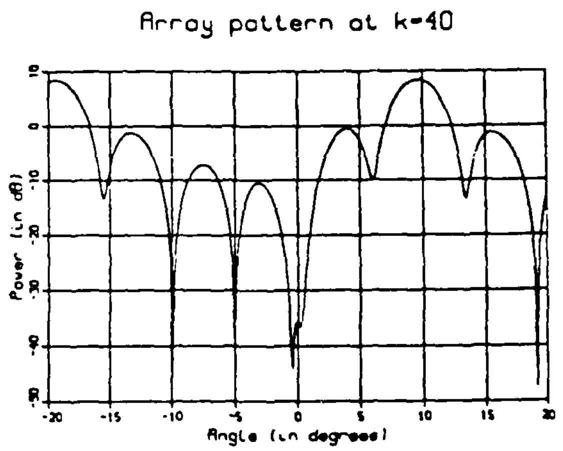
(a) SNIR for $\beta = 0.2$



(b) SNIR for $\beta = 0.05$

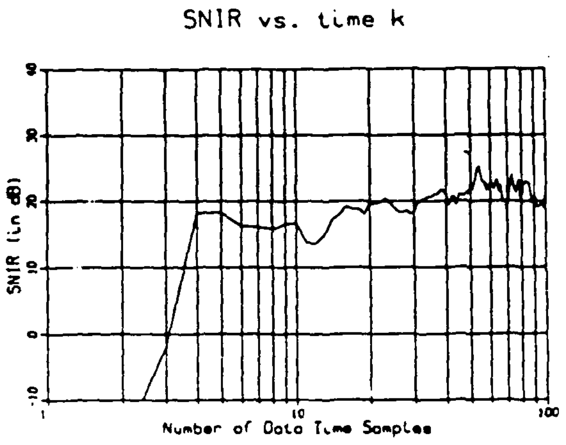


(c) Ant. pattern for $\beta = 0.05$

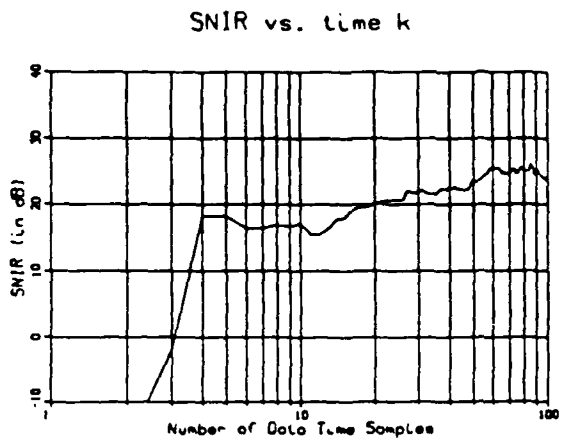


(d) Ant. pattern for $\beta = 0.05$

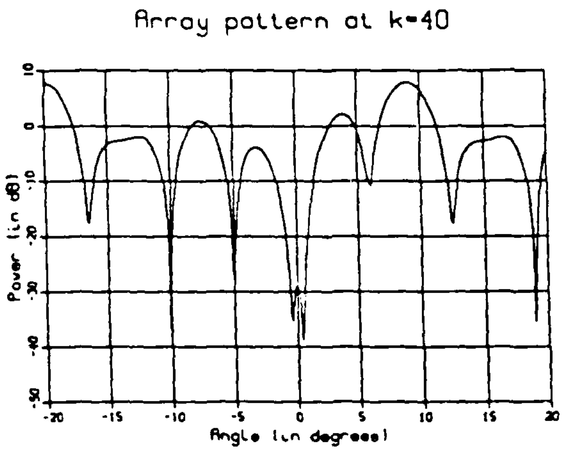
Figure 5: Multiple jammer scenario (case 8, 2 jammers only 0.5° apart)



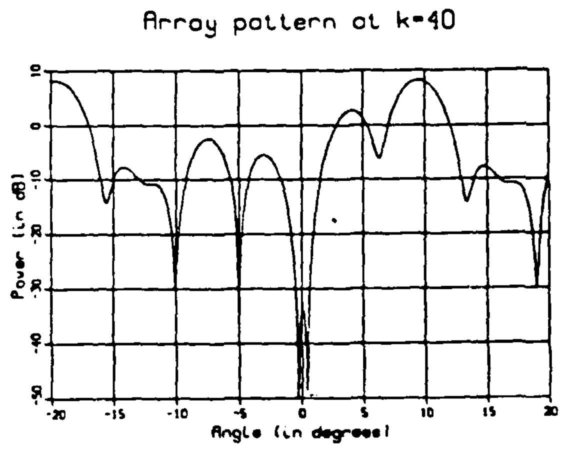
(a) SNIR for $\beta = 0.2$



(b) SNIR for $\beta = 0.05$

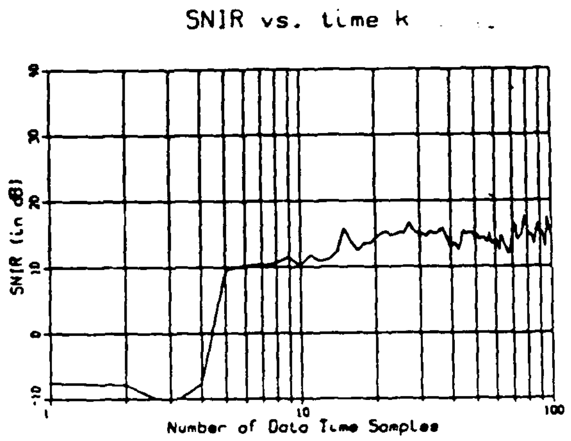


(c) Ant. pattern for $\beta = 0.05$

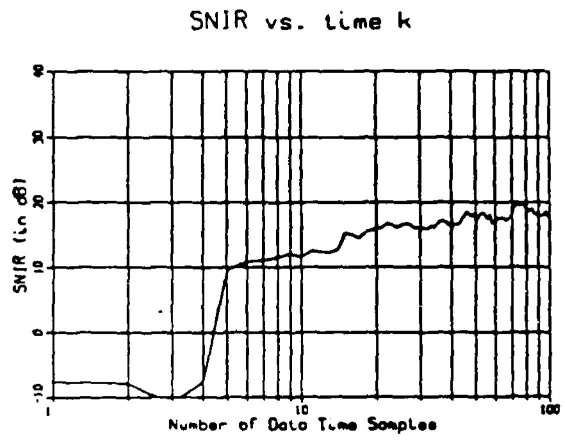


(d) Ant. pattern for $\beta = 0.05$

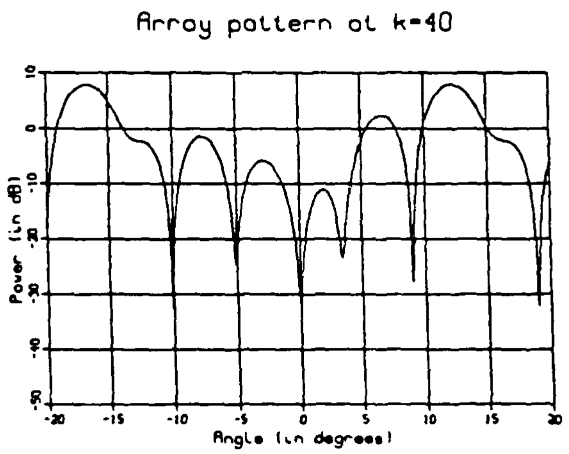
Figure 6: Multiple jammer scenario (case 12, with large $c = 8800$)



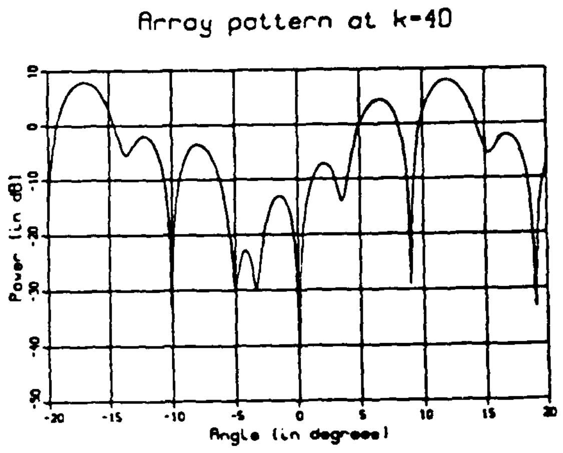
(a) SNIR for $\beta = 0.2$



(b) SNIR for $\beta = 0.05$

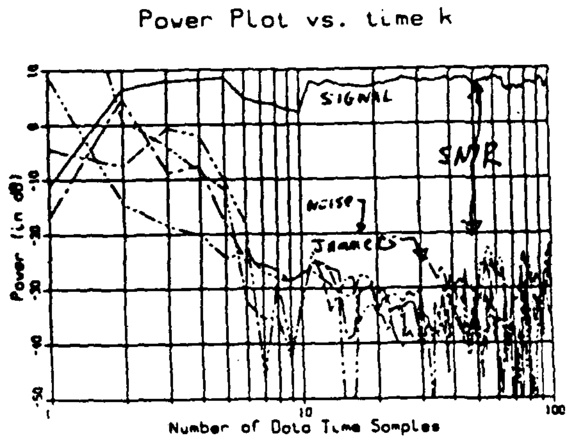


(c) Ant. pattern for $\beta = 0.05$

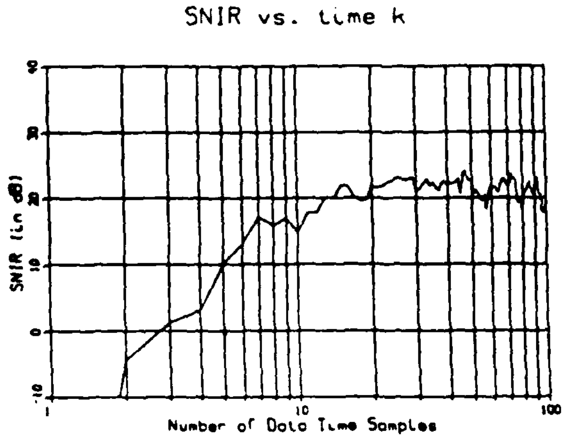


(d) Ant. pattern for $\beta = 0.05$

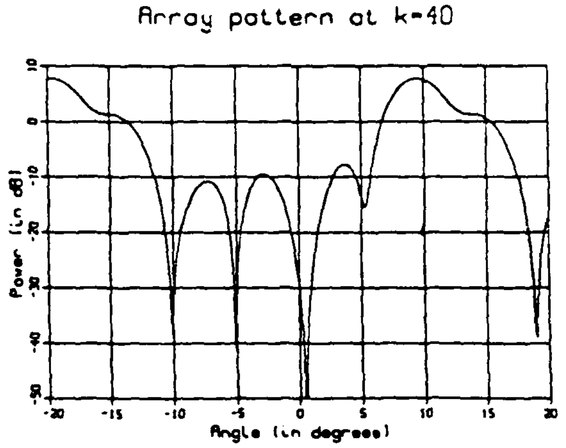
Figure 7: Multiple jammer scenario (case 13, mainbeam jammer)



(a) Power plot

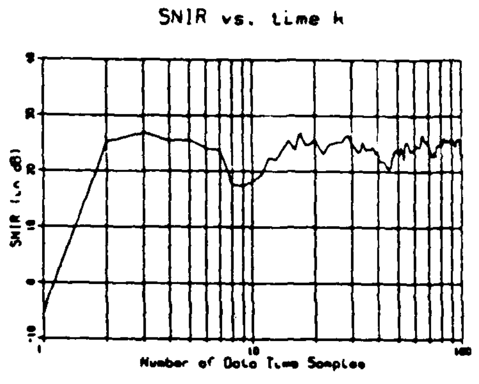


(b) SNIR plot

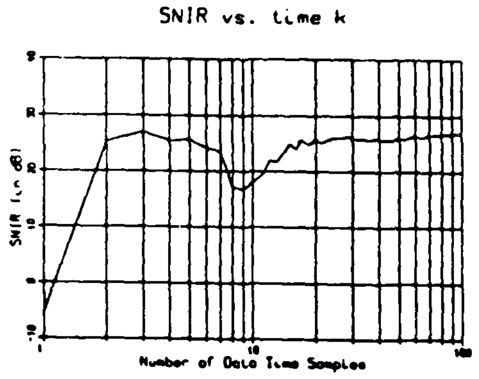


(c) Ant. pattern at $k = 40$

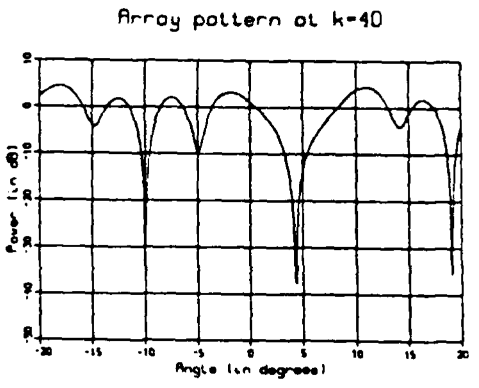
Figure 8: Strong jammer scenario (case 14, with large $C = 10^7$, $\beta = 0.2$)



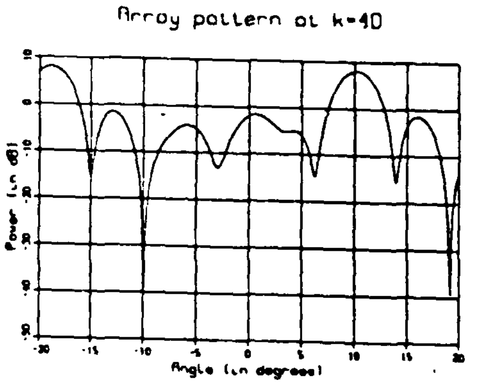
(a) SNIR for uniform weighting



(b) SNIR for exp. weighting



(c) Ant. pattern for uniform weighting



(d) Ant. pattern for exp. weighting

Figure 9: Effects of uniform ($\beta = 1/k$) or exponential ($\beta = 0.2$) weighting on the cancellation of a blinking jammer (off for 20 samples, on for 20, etc.)

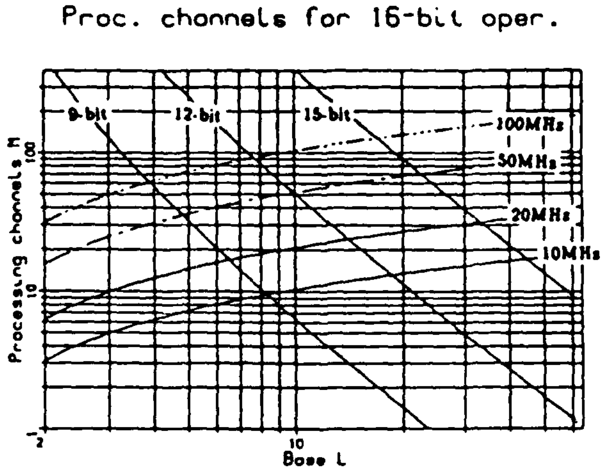


Figure 10: The number of channels M permissible for various A/D speeds and resolutions, as a function of the base L used

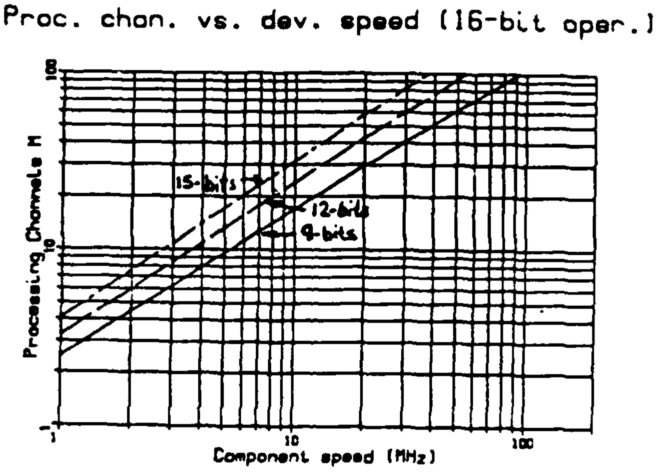


Figure 10: Maximum number of processing channels permissible when using A/Ds of various resolutions, as a function of A/D device speed.

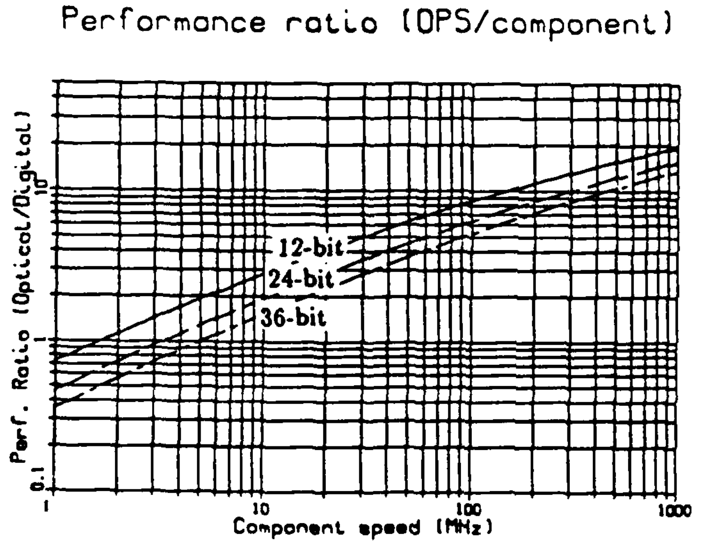


Figure 11: Optical vs. digital performance ratio as a function of component speed for 12-bit devices

CHAPTER 12:

DOCUMENTATION OF OUR AFOSR RESEARCH

12.1 AFOSR PUBLICATIONS

1. "Acousto-Optic Processor for Adaptive Radar Noise Environment Characterization", Applied Optics, 23, pp. 4303-4308, December 1984 (Goutzoulis, CASASENT, KUMAR).

2. "Frequency-Multiplexed Acousto-Optic Architectures and Applications", Applied Optics, 24, pp. 856-858, March 1985 (CASASENT).

3. "Fabrication and Testing of a Space and Frequency-Multiplexed Optical Linear Algebra Processor", OSA Topical Meeting on Optical Computing, pp. TuD7-1 - TuD7-4, March 1985 (CASASENT).

4. "Banded-Matrix High-Performance Algorithm and Architecture", Applied Optics, Vol. 24, pp. 1476-1480, 15 May 1985 (CASASENT, Taylor).

5. "Optical Linear Algebra Processors: Noise and Error-Source Modeling", Optics Letters, 10, pp. 252-254, June 1985 (CASASENT, Ghosh).

6. "A Factorized Extended Kalman Filter", Proc. SPIE, 564, pp. 119-130, August 1985 (Fisher, CASASENT, Neuman).

7. "Performance of Direct and Iterative Algorithms on an Optical Systolic Processor", Applied Optics, 24, pp. 3883-3892, 15 November 1985 (Ghosh, CASASENT, Neuman).

8. "Negative-Base Encoding in Optical Linear Algebra Processors", Applied Optics, *Rapid Communications*, Vol. 25, pp. 168-169, 15 January 1986 (Perlee, CASASENT).

9. "Optical Linear Algebra Processors: Architectures and Algorithms", Proc. SPIE, Vol. 614, pp. 153-164, January 1986 (CASASENT).

10. "Error-Source Effects in a High-Accuracy Optical Finite-Element Processor", Applied Optics, Vol. 25, pp. 966-975, 15 March 1986 (Taylor, CASASENT).

11. "Twos-Complement Data Processing for Improved Encoded Matrix-Vector Processors", Applied Optics, Vol. 25, pp. 956-961, 15 March 1986 (Taylor, CASASENT).

12. "Optical Linear Algebra Processor: Laboratory System Performance for Optimal Control Applications", Proc. SPIE, Vol. 639, pp. 68-75, March-April 1986 (CASASENT, Jackson).

13. "Bipolar Biasing in High-Accuracy Optical Linear Algebra Processors", Applied Optics, Vol. 25, pp. 1033-1035, 1 April 1986 (CASASENT, Perlee).

14. "Factorized Extended Kalman Filter for Optical Processing", Applied Optics, Vol. 25, pp. 1615-1621, 15 May 1986 (Fisher, CASASENT, Neuman).

15. "Optical Array Processor: Laboratory Results", Proc. SPIE, Vol. 700, *IOCC 1986*, Jerusalem, Israel, July 6-11, 1986 (CASASENT, Jackson, Vaerewyck).

16. "Space and Frequency-Multiplexed Optical Linear Algebra Processors: Fabrication and Initial Tests", Applied Optics, Vol. 25, pp. 2258-2263, 15 July 1986 (CASASENT, Jackson).

17. "Time and Space Integrating Optical Laboratory Matrix-Vector Array Processor", Proc. SPIE, Vol. 698, August 1986 (CASASENT, Riedl).

18. "Real-Time Optical Laboratory Linear Algebra Solution of Partial Differential Equations", Proc. SPIE, Vol. 698 August 1986 (CASASENT, Jackson).

19. "Laboratory Optical Linear Algebra Processor for Optimal Control", Optics Communications, Vol. 60, pp. 1-4, 15 October 1986 (CASASENT, Jackson).

20. "Linear Acousto-Optic Heterodyning Processors for Complex-Valued Data Processing", Proc. SPIE, Vol. 752, pp. 155-171, January 1987 (Pochapsky, CASASENT).

21. "Optical Linear Heterodyne Matrix-Vector Processor", Proc. SPIE, Vol. 886, pp. 158-170, January 1988 (Pochapsky, CASASENT).

22. "Direct Finite Element Solution on an Optical Laboratory Matrix-Vector Processor", Optics Communications, Vol. 65, No. 5, pp. 329-333, 1 March 1988 (CASASENT, Riedl).

23. "Optical Matrix-Vector Processing for Computational Fluid Dynamics", Proc. SPIE, Vol. 936, April 1988 (Perlee, CASASENT).

24. "Optical Matrix-Vector Processing for Finite Element Problems", Proc. SPIE, Vol. 939, pp. 2-11, April 1988 (Taylor, CASASENT).

25. "Reduced Sensitivity Algorithm for Optical Processors Using Constraints and Ridge Regression", Applied Optics, Vol. 27, pp. 1607-1611, 15 April 1988 (CASASENT, Ghosh).

26. "Factorized Extended Kalman Filter: Case Study Results", Applied Optics, Vol. 27, pp. 1877-1885, 1 May 1988 (Fisher, CASASENT, Neuman).

27. "Real-Time Optical Laboratory Solution of Parabolic Differential Equations", Applied Optics, Vol. 27, No. 14, pp. 2922-2925, 15 July 1988 (CASASENT, Jackson).

28. "Optical Laboratory Solution and Error Model Simulation of a Linear Time-Varying Finite Element Equation", Proc. SPIE, Vol. 977, August 1988, San Diego (Taylor, CASASENT).

29. "Variable Accuracy Optical Matrix/Vector Processors - Speed/Accuracy Tradeoffs", Proc. SPIE, Vol. 752, pp. 179-186, January 1987 (KUMAR, Carroll).

30. "Iterative Fourier Transform Phased Array Radar Pattern Synthesis", Proc. SPIE, Vol. 827, pp. 73-84, August 1987 (Carroll, KUMAR).

31. "Pattern Synthesis using Fourier Transforms", Optical Engineering, Submitted, October 1988 (Carroll, KUMAR).

32. "Application of Optical Processing to Adaptive Phased Array Radar", Proc. SPIE, Vol. 936, April 1988 (Carroll, KUMAR)

33. "Optical Systems for Digit-Serial Computation", Applied Optics, accepted, 1 February 1989 publication (Perlee, CASASENT).

34. "Discrete Steepest Descent Algorithms and their Realization on Optical Analog Processors", Proc. SPIE, Vol. 977, August 1988, San Diego (Pochapsky, CASASENT).

12.2 ORAL PRESENTATIONS

1. Sandia National Laboratories - Albuquerque, NM, "Optical Pattern Recognition and Optical Processing" (January 1985).

2. NASA Lewis - Cleveland, OH, "Optical Linear Algebra Processors (Systolic)" (February 1985).

3. George Washington University, - Washington, D.C., "Optical Linear Algebra for SDI" (March 1985).

4. OSA Topical Meeting on Optical Computing - Lake Tahoe, NV, "Fabrication and Testing of a Space and Frequency-Multiplexed Optical Linear Algebra Processor" (March 1985).

5. Eglin Air Force Base - Ft. Walton Beach, FL, "Optical Pattern Recognition and Kalman Filtering" (April 1985).

6. SPIE - San Diego, CA, "A Factorized Extended Kalman Filter" (August 1985).

7. NASA Langley Research Center, C.S. Colloquium - Hampton, VA, "Optical Array Processors" (November 1985).

8. SPIE - Los Angeles, CA, "Optical Linear Algebra Processors: Architectures and Algorithms" (January 1986).

9. Jet Propulsion Laboratory/NASA - Pasadena, CA, "Optical Linear Algebra and Pattern Recognition Processors" (January 1986).

10. SPIE - Orlando, FL, "Matrix-Vector Laboratory System and Application" (April 1986).

11. IBM, Federal Systems Division - Manassas, VA, "Optical Computing" (May 1986).

12. General Electric - Philadelphia, PA, "Adaptive Optical Processing" (May 1986).

13. Rockwell Corporation - Seal Beach, CA, "Optical Signal Processing" (May 1986).

14. Carnegie Mellon University, Professional Education Program - Pittsburgh, PA, "Optical Signal Processing" (June 1986).

15. IOCC Conference - Jerusalem, Israel, "Laboratory Optical Matrix-Vector Processors" (July 1986).

16. SPIE Conference - San Diego, CA, "Time and Space Integrating Optical Laboratory Matrix-Vector Array Processor" (August 1986).

17. SPIE Conference - San Diego, CA, "Real-Time Optical Laboratory Linear Algebra Solution of Partial Differential Equations" (August 1986).

18. Carnegie Mellon University, ECE Graduate Seminar - Pittsburgh, PA, "Optical Computing in ECE: 1986" (October 1986).

19. SPIE Conference - Los Angeles, CA, "Complex Data Handling in Analog and High-Accuracy Optical Linear Algebra Processors" (January 1987).

20. SPIE Conference - Los Angeles, CA, "Variable Accuracy Optical Matrix/Vector Processors - Speed/Accuracy Tradeoffs" (January 1987).

21. Teledyne Brown Engineering - Huntsville, AL, "Optical Signal Processing" (April 1987).

22. SPIE Conference - San Diego, CA, "Iterative Fourier Transform Phased Array Radar Pattern Synthesis" (August 1987).

23. McMaster University - Hamilton, Ontario, "Acousto Optic Signal Processing" (September 1987).

24. SPIE Conference - Los Angeles, CA, "Optical Linear Heterodyne Matrix-Vector Processor" (January 1988).

25. SPIE Conference - Orlando, FL, "Optical Matrix-Vector Processing for Computational Fluid Dynamics" (April 1988).

26. SPIE Conference - Orlando, FL, "Application of Optical Processing to Adaptive Phased Array Radar" (April 1988).

27. SPIE Conference - Orlando, FL, "Optical Matrix-Vector Processing for Finite Element Problems" (April 1988).

28. SPIE Conference - San Diego, CA, "Optical Laboratory Solution and Error Model Simulation of a Linear Time-Varying Finite Element Equation" (August 1988).

29. SPIE Conference - San Diego, CA, "Discrete Steepest Descent Algorithms and their Realization on Optical Analog Processors" (August 1988).

12.3 STUDENTS SUPPORTED AND DEGREES AWARDED UNDER AFOSR SUPPORT

1. Steven Riedl: M.S., "An Electronic Support System for Optical Matrix-Vector Processors", March 1986.

2. James Jackson: Ph.D., "Development and Performance Evaluation of an Acousto-Optic Linear Algebra Processor", January 1986.

3. Christopher Carroll: Ph.D., "Application of Optical Signal Processing to Adaptive Phased Array Radar", May 1987.

4. Caroline Perlee: Ph.D., "Optical Computing Systems for Linear Algebraic Processing and On-Line Arithmetic Algorithms", October 1988.

5. Bradley K. Taylor: Ph.D., "Simulation and Test of an Optical Matrix-Vector Processor", August 1988.

6. Eugene Pochapsky: Ph.D., "Acousto-Optic Linear Algebra Processors", November 1988.